VNF placement in 5G Networks using AI/ML

Masoud Taghavian

▶ **To cite this version:**

Masoud Taghavian. VNF placement in 5G Networks using AI/ML. Computer Science [cs]. Ecole nationale supérieure Mines-Télécom Atlantique, 2024. English. NNT : 2024IMTA0421 . tel-04845476

HAL Id: tel-04845476
https://theses.hal.science/tel-04845476v1

Submitted on 18 Dec 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

COLLEGES SCIENCES
BRETAGNE POUR L'INGENIEUR
LOIRE ET LE NUMERIQUE

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

THÈSE DE DOCTORAT DE

L'ÉCOLE NATIONALE SUPÉRIEURE
MINES-TÉLÉCOM ATLANTIQUE BRETAGNE
PAYS DE LA LOIRE – IMT ATLANTIQUE

ÉCOLE DOCTORALE N° 648
Sciences pour l'Ingénieur et le Numérique
Spécialité : *Sciences et technologies de l'information et de la communication / Informatique*

Par
Masoud TAGHAVIAN

VNF Placement in 5G Networks using AI/ML

Thèse présentée et soutenue à IMT Atlantique, Rennes, le 25/09/2024
Unité de recherche : IRISA
Thèse N° : 2024IMTA0421

Rapporteurs avant soutenance :

Emmanuel CHAPUT	Professeur, INP-ENSEEIHT
Yacine GHAMRI-DOUDANE	Professeur, La Rochelle University

Composition du Jury :

Président:	Emmanuel CHAPUT	Professeur, INP-ENSEEIHT (Présent)
Examinateurs :	Yacine GHAMRI-DOUDANE	Professeur, La Rochelle University (En Visio)
	Christelle CAILLOUET	Maître de Conférence, University of Côte d'Azur (En Visio)
	Yassine HADJADJ-AOUL	Professeur, University of Rennes (Présent)
Dir. de thèse :	Geraldine TEXIER	Professeure, IMT Atlantique and IRT-BCOM (Présent)
Encadrant de thèse :	Philippe BERTIN	Docteur, Orange and IRT-BCOM (Présent)

Acknowledgement

First of all, I would like to appreciate my director of the thesis Pr. Geraldine Texier in IMT Atlantique, my superviser Pr. Yassine Hadjadj-Aoul in University of Rennes 1, and my supervisor Dr. Philippe Bertin in IRT bcom, who have never hesitated for their helps, and their supports, in every step of my work, from their enlightening advices to the writings and the publications.

I would also like to express my gratitude to the jury members for undertaking the process of the review of my work. Their highly appreciated comments have largely contributed to the improvement of the quality of this work.

I would also need to mention my appreciation to Carole Bonan, Olivier Choisy, and all of the colleagues and friends in the Network and Connectivity Lab in IRT bcom, who I had the opportunity to exchange and learn during my PhD.

I would also want to thank Nicolas Huin, for his remarkable contribution in the optimization modeling, his technical helps in latex, writings and the publication of our journal paper.

Dedication

I dedicate my thesis to the great people who have the courage to question, to shake, and to enlighten, and then to decide to find solace in living with the questions rather than the alluring answers in the stories and the fairy tales.

I also dedicate my thesis to the outstanding people who fight for freedom all over the world, especially in my country Iran.

Table of Contents

Acronyms

AI Artificial Intelligence

ML Machine Learning

RL Reinforcement Learning

DDPG Deep Deterministic Policy Gradient

DRL Deep Reinforcement Learning

GNN Graph Neural Network

RGCN Relational Graph Convolutional Neural Network

BnB Branch and Bound

UCS Uniform-Cost Search

VM Virtual Machine

LP Linear Programming

ILP Integer Linear Programming

MIQCP Mixed Integer Quadratically Constrained Programming

MILP Mixed Integer Linear Programming

CG Column Generation

GA Genetic Algorithm

PSO Particle swarm optimization

ACO Ant Colony Optimization

TS Tabu Search

BFS Breadth-First Search

DFS Depth-First Search

MANO Management and Orchestration

eMBB Enhanced Mobile Broadband

mMTC Massive Machine Type Communication

URLLC Ultra-reliable and Low Latency Communications

NFV Network Function Virtualization

MEC Multi-access Edge Computing

CC Cloud Computing

DC Data Center

SDN Software-Defined Networking

NFVO Network Functions Virtualization Orchestrator

NS Network Service

PNF Physical Network Function

CNF Cloud-native Network Function

QoS Quality of Service

E2E End-to-End

SA Service Acceptance

SLA Service Level Agreement

SFC Service Function Chaining

SG Service Graph

VNS Virtual Network Service

SN Substrate Network

VIM Virtualized Infrastructure Manager

VNF Virtual Network Function

VNFC Virtual Network Function Component

VDU Virtual Deployment Unit

NF Network Function

VNE Virtual Network Embedding

VL Virtual Link

VNFM Virtual Network Function Manager

VNFPP VNF Placement Problem

VNFFG VNF Forwarding-Graph

LIFO Last In First Out

PI Parallel Integrated

CN Core Network

RAN Radio Access Network

CRAN Cloud Radio Access Network

AV Autonomous Vehicles

CDN Content Delivery Network

IoT Internet of Things

AR Augmented Reality

MNO Mobile Network Operator

API Application Programming Interface

ONOS Open Network Operating System

OS Operating System

ONF Open Networking Foundation

TSP Telecommunications Service Provider

COTS Commercial Off-The-Shelf

ETSI European Telecommunications Standards Institute

3GPP The 3rd Generation Partnership Project

GPGPU General-Purpose Graphics Processing Unit

FPGA Field Programmable Gate Arrays

DSP Digital Signal Processing

DPI Deep Packet Inspection

IPS Intrusion Prevention System

FF First-Fit

BF Best-Fit

RF Random-Fit

EBF Enhanced Best-Fit

ABO A* Bandwidth Optimized

DBO DFS Bandwidth Optimized

FABO Fair A* Bandwidth Optimized

Résumé en Français

En 2023, près de 60 % du trafic Web provient des 6 milliards d'abonnés mobiles des réseaux cellulaires du monde entier [1]–[4]. Cela met en évidence le rôle vital du Mobile Network Operators (MNOs) dans la fourniture de connexions, de services et de contenus à des utilisateurs toujours plus nombreux et aux attentes de plus en plus élevées.

L'évolution continue des technologies de télécommunication a conduit à des attentes de performances plus élevées. Bien qu'il puisse sembler logique d'attendre des performances 10 fois supérieures de la 5G par rapport à la 4G, similaires à l'augmentation observée de la 3G à la 4G, en réalité chaque technologie a des limites. Pour répondre à ces attentes, les améliorations progressives peuvent ne pas suffire. Au lieu de cela, la 5G adopte des innovations plus prometteuses, s'éloignant des technologies précédentes. Ces attentes et exigences de performances sont motivées par divers cas d'utilisation pionniers nécessitant soit une plus grande bande passante, une meilleure qualité de service ou davantage de connexions prises en charge. Ces exigences sont illustrées dans trois catégories principales, selon l'UIT [5], qui inclut également certains cas d'utilisation.

— *Enhanced Mobile Broadband (eMBB)*: eMBB préoccupations concernant la fourniture de la bande passante requise par plusieurs cas d'utilisation qui nécessitent la diffusion de vidéos 3D UHD comme dans Augmented Reality (AR). De plus, le remplacement des connexions filaires par des alternatives sans fil capables de fournir au moins la même quantité de bande passante, pour de nombreux cas d'utilisation différents allant des verticaux 5G via des connexions Ethernet en cuivre, ou des connexions Internet à domicile en fibre optique, voire des connexions backhole entre le réseau central et les réseaux périphériques, est une autre ambition qui est suivie dans cette catégorie. Le besoin mentionné découle du coût ennuyeux de l'établissement et de la maintenance des connexions filaires.

— *Massive Machine Type Communication (mMTC)*: mMTC vise à garantir la couverture d'un nombre massif d'objets connectés dans les villes intelligentes et les systèmes Internet of Things (IoT), des réseaux intelligents dans le secteur de l'énergie aux capteurs/actionneurs dans l'agriculture, les services d'eau, les soins de santé, etc.

— *Ultra-reliable and Low Latency Communications (URLLC)*: URLLC se concentre

sur la fourniture des exigences strictes de latence et de fiabilité, qui sont rencontrées dans des cas d'utilisation tels que les véhicules intelligents et autonomes, les systèmes de transport intelligents, la chirurgie à distance, etc.

Notez qu'un cas d'utilisation spécifique représenté dans une catégorie comme les véhicules autonomes dans URLLC, en plus d'exiger une latence et une fiabilité strictes, a des exigences de bande passante et de connectivité, qui sont les principaux sujets dans eMBB et mMTC respectivement. En fait, un cas d'utilisation est identifié dans une catégorie, lorsqu'il touche aux limites de la technologie dans cette catégorie spécifique (c'est-à-dire que ses autres exigences sont réalisables par la technologie actuelle).

Les attentes ambitieuses des réseaux de cinquième génération ne peuvent pas être satisfaites simplement en améliorant la technologie des réseaux actuels et de la génération précédente. Pour se conformer à ces exigences, une révolution continue a commencé au cours de la dernière décennie sur les technologies de télécommunication. Plusieurs nouveaux paradigmes ont émergé comme Software-Defined Networking (SDN) et Network Function Virtualization (NFV), qui ont démontré un potentiel élevé dans la création de réseaux modernes flexibles et évolutifs.

Les principaux opérateurs de réseau ont adopté le SDN et le NFV comme solution pour des réseaux ultra-flexibles qui répondent aux diverses attentes des réseaux 5G. Le découplage des données et du plan de contrôle dans le SDN et l'utilisation des technologies de cloud computing pour virtualiser, orchestrer et faire évoluer divers types de fonctions réseau afin de réaliser des services dans le NFV sont en effet inévitables dans les réseaux modernes [6].

Nous avons généralement deux types d'opérations pour réaliser un réseau : la transmission des paquets et le traitement des paquets. Pour présenter SDN et NFV en un mot, SDN est apparu comme une solution pour faciliter la transmission des paquets, tandis que NFV comme un remède pour aider au traitement des paquets. Notez que toute l'histoire a pour but de se débarrasser des solutions monolithiques de type boîte noire spécifiques aux fournisseurs qui étaient presque impossibles à maintenir et à mettre à jour afin de répondre aux exigences de télécommunication en constante évolution. La nécessité de décomposer les opérations réseau complexes et de disposer de solutions standard et ouvertes, suscite l'intérêt pour le SDN et le NFV.

Auparavant, les routeurs traditionnels étaient responsables de décider du routage des paquets de manière distribuée. Cette décision était prise à l'aide d'une table de routage au sein d'un routeur, et chaque routeur était responsable de la création et de la mise à jour

de cette table en découvrant et en surveillant en permanence la topologie du réseau et la capacité de chaque lien réseau en fonction de sa charge de trafic réelle. La complexité de la synchronisation entre ces routeurs distribués dans une charge de trafic en constante évolution d'une part, et le fait d'avoir différents routeurs spécifiques à chaque fournisseur avec des algorithmes de routage propriétaires d'autre part, créent de gros obstacles pour que la transmission des paquets réponde aux exigences strictes de QoS.

Le SDN [7] est apparu comme un remède pour réduire la complexité mentionnée en centralisant le contrôle de la transmission des paquets. Le SDN remplace les routeurs par de simples commutateurs, et un contrôleur centralisé est chargé de surveiller le réseau et de mettre à jour les tables de routage des commutateurs de manière ouverte et standard.

Le concept de centralisation du contrôle des flux de paquets remonte aux tout premiers débuts d'Internet [8]. Autrefois, l'évolutivité était de loin plus importante que les exigences de QoS récemment apparues. Par conséquent, une approche distribuée était préférée à une approche centralisée dans le passé. Aujourd'hui, tout a changé. Les progrès dans le cloud computing, l'IA et la communauté open source ont changé de nombreux domaines, et les télécommunications ne font pas exception, au point qu'un contrôle centralisé semble désormais plus approprié.

Une architecture détaillée peut être trouvée dans [7], mais Open Networking Foundation (ONF) représente une architecture plus concrète de SDN avec ses trois couches principales dans [9]. La couche d'application demande les services réseau requis via une API en direction du nord vers le contrôleur SDN, et le contrôleur surveille et transmet les règles aux commutateurs réseau via une API en direction du sud. La couche d'infrastructure ou couche de données est composée des commutateurs qui sont responsables de la transmission des paquets à travers le réseau.

NFV est apparu pour mettre en logiciel et virtualiser les fonctions réseau, qui étaient auparavant fournies par des périphériques matériels dédiés, pour permettre leur exécution dans des serveurs à volume élevé standard de l'industrie Commercial Off-The-Shelf (COTS) [10]. Les principaux obstacles liés notamment à la maintenance, à la mise à jour, à la mise à l'échelle et à la migration du Physical Network Functions (PNFs) d'une part, et les avancées dans les technologies de virtualisation et le cloud computing d'autre part ont poussé l'écosystème des télécommunications à évoluer vers NFV sans hésitation. NFV, non seulement surmonte les obstacles mentionnés, mais réduit également les coûts en partageant les ressources, et nous conduit vers le rêve de l'automatisation et des réseaux sans contact.

Bien que la NFV apporte de nombreux avantages, elle présente plusieurs défis sans précédent. Tout d'abord, les nouveaux VNF logiciels doivent être au moins aussi performants, fiables et sécurisés que leurs équivalents matériels. Nous devons prendre en compte que la transition prendrait du temps, donc la compatibilité et la coexistence avec des PNF dédiés sont inévitables. La NFV est livrée avec les fonctions responsables de la gestion, de l'orchestration et finalement de l'automatisation des VNF. L'intégration, l'interopérabilité et la mise en place d'API standard entre l'orchestrateur et les VNF doivent également être abordées.

Un service réseau est traditionnellement composé d'un ensemble de PNF (par exemple des routeurs, des pare-feu, des équilibreurs de charge, etc.) liés entre eux pour fournir un utilitaire réseau. Avec l'avènement de la NFV, un service réseau est représenté sous la forme d'un graphe logique de Virtual Network Functions (VNFs) alignés ensemble via le Virtual Links (VLs) selon les spécifications établies par l'ETSI [11]. Un VNF est considéré comme le plus petit composant du point de vue du réseau, et il peut lui-même être décomposé en un graphe de Virtual Network Function Components (VNFCs) qui sont connectés à VLs. Un VNFC est considéré comme le plus petit composant du point de vue des ressources, qui est déployé sur un Virtual Deployment Units (VDUs).

La mise en œuvre de la décomposition de service réseau proposée est hors de portée des spécifications ETSI, mais pour citer certaines des technologies les plus populaires qui sont fréquemment choisies pour le développement, nous pouvons sélectionner MVC comme modèle de conception de logiciel [12], Microservices comme architecture logicielle [13], Docker comme gestionnaire de conteneurs [14] et Kubernetes comme système d'orchestration de conteneurs [15].

Pour réaliser les architectures SDN et NFV proposées, de nombreux outils existent comme Cloud Computing (CC), Multi-access Edge Computing (MEC), Serverless Computing (SC), les outils d'orchestration de conteneurs, etc., ainsi que des frameworks et des spécifications comme ETSI Management and Orchestration (MANO), Service Function Chaining (SFC), Network Slicing, etc.

De plus, de nombreuses solutions sont proposées et soutenues par la communauté open source, les principaux opérateurs de réseau et les fournisseurs d'équipements réseau. Certaines d'entre elles sont répertoriées comme suit:

— *Open Source MANO (OSM)* est un projet hébergé par l'ETSI visant à développer une pile logicielle Open Source NFV Management and Orchestration (MANO) alignée sur ETSI NFV [16].

— *Open Network Operating System (ONOS)* est proposé par la Linux Foundation comme contrôleur SDN sous la forme d'un système d'exploitation distribué open source évolutif, hautes performances et haute disponibilité. Notez qu'un système d'exploitation distribué haute disponibilité pour un contrôleur SDN peut sembler en contraste avec le concept de centralisation dans le SDN, mais nous devons être conscients que nous avons logiquement un contrôleur SDN et ses plusieurs répliques synchronisées pour des raisons de disponibilité.

— *Open Network Automation Platform (ONAP)* est un framework d'orchestration et d'automatisation open source hébergé par la Linux Foundation [17].

— *OpenDaylight* est un projet qui a été lancé comme une démonstration de concept et un prototype de SDN et NFV, puis il a été proposé comme un produit pour la surveillance, l'automatisation et la mise à l'échelle des réseaux de différentes tailles avec l'aide et le soutien de la fondation Linux, Red-Hat, Orange, Ericsson, Cisco, etc. [18]

— *OpenAirInterface (OAI)* est une plate-forme ouverte d'expérimentation et de prototypage créée par le département des communications mobiles d'EURECOM pour permettre l'innovation dans le domaine des réseaux et des communications mobiles/sans fil [19].

— *OpenFlow* est choisi comme API standard pour connecter le contrôleur SDN aux commutateurs réseau [20].

— *Open vSwitch (OVS)* est une implémentation open source d'un commutateur virtuel distribué entièrement compatible avec l'API OpenFlow.

Le placement des services de réseau virtualisés représente l'une des étapes les plus cruciales du SDN et du NFV. Il aborde l'allocation de ressources physiques pour les services réseau, exigeant des ressources hétérogènes avec diverses contraintes SLA. Une chose qui rend le problème de placement intéressant à étudier est sa complexité. Dans [21], le problème de placement est prouvé comme étant NP-complet en réduisant le problème de bin packaging (un problème NP-complet bien connu) à celui-ci.

Le placement est rencontré dans de nombreux domaines de recherche et cas d'utilisation dans les réseaux 5G allant du placement VNFFG et du découpage réseau à la virtualisation du CN, du CDN, de l'IoT et plus encore. De plus, il est nécessaire pendant toute la durée de vie des services réseau, depuis le placement initial du service dans le réseau substrat jusqu'à sa mise à l'échelle, sa migration et même sa destruction.

Le placement des services est effectué selon de nombreux objectifs dans la littérature. Ces objectifs sont abordés comme un problème d'optimisation, soumis à diverses contraintes, notamment les ressources, les exigences de QoS et d'autres contraintes qui peuvent être spécifiques à l'objectif et au contexte. Les optimisations des ressources, les optimisations de QoS (par exemple, latence, disponibilité, fiabilité, sécurité), la consommation d'énergie, le rapport coût/revenu (R2C) et l'AS font partie des objectifs bien étudiés. Nous nous intéressons particulièrement à l'AS. L'AS est définie comme le nombre maximal de services qui peuvent être placés avec succès sur le réseau sous réserve des ressources et des exigences de QoS.

Deux catégories générales prises en compte pour le problème de placement sont les placements hors ligne et en ligne. Dans un scénario de placement hors ligne, nous connaissons à l'avance toutes les demandes de placement de services, tandis que dans un scénario de placement en ligne, nous n'avons aucune information sur les demandes de placement futures et nous devons effectuer un placement pour la demande actuelle dès son arrivée. Nous pouvons également avoir des restrictions de temps pour effectuer des placements dans un placement en ligne, ce qui fait de l'évolutivité une exigence importante qui doit être prise en compte [22]. Dans notre thèse, nous étudions le problème de placement dans un contexte en ligne.

Nous abordons certains des défis liés au placement des services réseau sur les réseaux substrats en NFV, en tenant compte des ressources et des exigences de Quality of Service (QoS). Nos contributions peuvent être résumées comme suit:

— *L'état de l'art*: Nous fournissons une étude des travaux connexes dans ce domaine, et nous présentons une revue de la littérature des approches de placement dans de nombreux cas d'utilisation NFV, suivant différents objectifs et contraintes. De plus, nous explorons les travaux connexes traitant du placement dans des contextes spécifiques comme la bande passante limitée dans le chapitre 4 et les ressources de nœuds limitées dans le chapitre 5.

— *Placement de services réseau avec bande passante limitée*: Le respect des exigences de QoS lorsque la bande passante réseau est limitée représente l'un des défis dans ce domaine de recherche. Nos publications de [23]–[26] sont les résultats de nos études dans ce contexte. Le chapitre 4 est consacré à ce problème, dans lequel nous couvrons les éléments suivants:

  — Modèle d'optimisation

  — Définition du problème comme approche de recherche Branch and Bound (BnB)

  - Application de l'algorithme A*
  - Basé sur les nœuds ou basé sur les liens BnB
  - Distribution équitable de la bande passante
  - Défragmentation du réseau
- *Placement de services réseau avec des ressources de nœuds limitées*: Le placement en ligne des services réseau, exigeant des exigences strictes de latence End-to-End (E2E), sur les réseaux de périphérie, avec des ressources informatiques restreintes présente un problème difficile qui mérite d'être étudié. Nos publications de [27] sont le résultat de nos travaux dans cette direction. Le chapitre 5 est consacré à ce problème, dans lequel nous couvrons les éléments suivants:
  - Modèle d'optimisation
  - Définition du problème comme approche de recherche BnB
  - Application de plusieurs fonctions de coût et stratégies de recherche
  - Éviter l'optimisation de la latence
  - Distribution équitable des ressources du nœud
- *Outil Network Service Placer (NSPlacer)*: Dans la section Annexe, nous démontrons notre outil *NSPlacer*, que nous avons développé pour réaliser des évaluations et des comparaisons entre différents algorithmes de placement, y compris notre approche proposée dans [23]. De plus, il fournit une infrastructure permettant à la communauté de recherche d'évaluer ses propres méthodes de placement.

Une partie importante des algorithmes de placement implique les résolutions exactes, en formulant le problème comme Integer Linear Programming (ILP) ou Mixed Integer Linear Programming (MILP), et en essayant de le résoudre via des solveurs et des optimiseurs.

Dans [28], les auteurs visent à optimiser le placement de services multi-contraints, en considérant la bande passante et la latence des liens, et différentes ressources système pour les nœuds sur un graphe de topologie de réseau en étoile de type edge-core. Leur objectif est de maximiser le nombre de demandes de service placées, d'abord en proposant une résolution ILP, puis une heuristique. Dans un effort pour assurer une répartition équitable des ressources, ils ont considéré le coût d'utilisation d'une ressource proportionnellement à la capacité restante du nœud ou du lien correspondant (c'est-à-dire que le coût d'utilisation d'une ressource sur un nœud/lien surchargé est bien supérieur à celui d'un nœud/lien sous-chargé). Les chercheurs ont ensuite démontré que cette légère modification se traduit

par une répartition plus équitable des ressources et conduit à une augmentation du nombre de services acceptés.

Malgré l'optimalité et la clarté des méthodes basées sur ILP, elles souffrent de problèmes d'évolutivité. Notez que l'exécution d'une résolution exacte nécessite un long temps de traitement jusqu'à l'obtention d'une solution optimale, et elle ne peut être utilisée que pour les petites topologies de réseau. Cette limitation constitue un obstacle important dans les scénarios de placement en ligne avec des contraintes en temps réel.

En ce qui concerne l'évolutivité des approches, les heuristiques sont souvent préférées. Best-Fit ou Decreasing First-Fit est l'une des heuristiques de placement les plus connues et les plus efficaces, qui trie les nœuds et place le VNF sur le nœud contenant la quantité maximale de ressources disponibles de manière itérative. [29] explore les algorithmes heuristiques évolutifs et rentables, en particulier l'algorithme Best-Fit, et propose un algorithme de graphe à plusieurs étapes pour trouver des solutions quasi optimales de manière évolutive. Ils ont supposé que les demandes de service arrivent de manière aléatoire, en considérant une durée de vie aléatoire pour chaque service. Les services sont composés de 3 VNF, nécessitant des unités aléatoires de ressources pour les VNF et les VL. La durée de simulation est fixée à une unité de temps pour chaque exécution, et ils étudient le temps d'exécution (y compris l'évolutivité des algorithmes Best-Fit et Multi-Stage), le taux d'acceptation et le coût moyen du placement en termes de ressources attribuées.

Bien que conçues pour obtenir une solution dans un délai raisonnable, les heuristiques n'offrent pas de garanties de qualité et le risque de rester bloqué dans des optimaux locaux est extrêmement élevé.

Les algorithmes méta-heuristiques et évolutifs sont une autre direction pour résoudre le problème de placement.

Dans [30], une approche basée sur un algorithme génétique est proposée. Ils ont comparé leurs résultats à l'algorithme Best-Fit comme algorithme de base, et ils ont obtenu des taux d'acceptation des demandes plus élevés, résultant d'une utilisation plus efficace des ressources par rapport à un algorithme gourmand de base avec un objectif d'optimisation similaire.

Bien qu'ils soient puissants pour échapper aux optimaux locaux, ils souffrent d'imprévisibilité, notamment en termes de temps d'exécution, ce qui est extrêmement important dans le placement en ligne.

Les mécanismes de recherche d'IA avancés et les développements récents dans les techniques d'apprentissage automatique, et en particulier Deep Reinforcement Learning

(DRL), ont attiré beaucoup d'attention en raison de leur potentiel prometteur.

Les auteurs de [31] explorent le potentiel de DRL pour maximiser le nombre de services acceptés, en satisfaisant les exigences de QoS. Ils montrent que leur approche DRL peut apprendre la relation non linéaire entre les métriques de QoS et le trafic, afin de pouvoir trouver de meilleurs placements, ce qui se traduit par des services plus acceptés.

[32] essaie DRL avec Relational Graph Convolutional Neural Network (RGCN) pour maximiser l'acceptation du service. Dans [33], une approche DRL utilisant des réseaux Q Double Deep est proposée pour réduire le taux de rejet des demandes de service, tout en améliorant l'évolutivité du système grâce à une formation hors ligne. Une approche Q-Learning est étudiée dans [34], essayant d'établir un compromis entre évolutivité et optimalité afin de maximiser l'avantage du fournisseur de services. Ils pourraient obtenir des solutions quasi optimales dans un délai relativement court, tout en tenant compte de l'équilibre de charge du réseau.

Notre travail s'inscrit dans cette dernière catégorie, découvrant le potentiel des stratégies de recherche d'IA, en particulier A^*. Il convient de préciser la position de notre travail parmi les approches basées sur le ML. Dans notre contribution, nous sommes capables de trouver des placements optimaux (dans le pire des cas quasi optimaux), tout en visant à être aussi évolutifs que possible. Alors que dans les approches basées sur le ML, ces méthodes sont efficaces en termes de calcul après la phase d'apprentissage, l'objectif étant d'obtenir des placements aussi proches que possible des résultats optimaux et quasi-optimaux en exploitant de nouvelles approches d'apprentissage. Ce sont les différences importantes entre notre travail et les approches récentes basées sur le ML.

Le placement en ligne des services réseau, exigeant des exigences strictes de latence E2E, sur les réseaux périphériques, avec des ressources de calcul restreintes compte tenu de la localisation de l'utilisateur final représente un problème difficile qui mérite d'être étudié. C'est le sujet que nous essayons d'aborder dans ce chapitre.

Pour répondre à la latence requise des services réseau, nous ne pouvons pas les placer sur les clouds loin des utilisateurs finaux, où les ressources sont abondantes. Certains services doivent nécessairement être conservés à la périphérie du réseau, où les ressources sont plus rares.

En suivant notre objectif de SA, dans un placement hors ligne, bien que la complexité ne soit pas négligeable, par un modèle ILP bien défini, nous pouvons effectuer une optimisation pour trouver le nombre maximum de services acceptés (*a.k.a.* taux d'acceptation). Dans le cadre d'un placement en ligne, même si nous ne connaissons pas à l'avance les demandes

de service (et ne pouvons donc pas optimiser directement le nombre de services acceptés), nous pouvons nous assurer que le placement de la demande de service actuelle augmente les chances d'acceptation des demandes de service futures. Cela peut être réalisé en optimisant une fonction de coût cohérente avec notre objectif.

Nous respectons la contrainte anti-affinité, dans laquelle nous n'acceptons pas de placer deux VNF du même service sur le même nœud de réseau.

Bien que le problème du placement ait été étudié dans plusieurs domaines connexes, à notre connaissance, le placement des demandes de service avec une latence stricte E2E sur le réseau périphérique ayant des ressources de nœud limitées, compte tenu de la localisation de l'utilisateur et de l'anti-affinité, n'est pas encore abordé.

En général, les approches de placement proposées tentent d'optimiser le coût des ressources ou les exigences QoS telles que la bande passante et la latence. Un service réseau nécessite des ressources de nœud (*a.k.a.* ressources informatiques) pour déployer son VNFs et des ressources de liaison (*a.k.a.* ressources réseau) pour réaliser son VLs. En supposant que nous ne partagions pas le VNF déjà placé (comme dans [35]), et que nous devons fournir ce qui est requis comme ressources de nœud pour déployer le VNFs, l'optimisation consiste à placer le VNFs aussi près que possible de l'emplacement de l'utilisateur et de l'autre VNFs, afin de minimiser l'utilisation globale de la bande passante et/ou la latence E2E. Nous montrerons que cette approche d'optimisation pourrait avoir un effet dévastateur sur les réseaux périphériques avec des ressources de nœud limitées et doit être évitée. Nous verrons qu'elle peut conduire à épuiser les ressources à proximité de l'emplacement de l'utilisateur, ce qui entraîne un rejet de service. Ainsi, nous ne suivons pas une approche d'optimisation de la bande passante/latence, cependant, nous les considérons comme des contraintes qui doivent être respectées.

Nous modélisons le problème comme un ILP et résolvons le modèle pour obtenir l'optimum global (qui ne peut être atteint que dans un scénario hors ligne). Nous proposons une approche en ligne basée sur BnB, capable d'obtenir des résultats optimaux et quasi-optimaux selon plusieurs fonctions de coût et stratégies de recherche. Nous explorons également plusieurs fonctions de coût et démontrons l'incohérence de l'optimisation de la bande passante et/ou de la latence, ainsi que les avantages d'un placement équitable, avec notre objectif de maximiser le Service Acceptance (SA).

Le *caractère équitable* de notre approche peut s'expliquer de deux manières. D'une part, nos améliorations obtenues sont dues à la répartition équitable des ressources dans nos placements. D'autre part, notre approche fait un compromis équitable entre le fournisseur

de services et le fournisseur de réseau/cloud. Le fournisseur de services préfère que la latence réalisée de ses services soit la plus faible possible (idéalement zéro), tandis que le fournisseur de réseau préfère qu'elle soit la plus élevée possible (maximum acceptable), ce qui lui permet de pousser le placement des services de la périphérie vers les clouds où les ressources sont abondantes et moins chères. La latence maximale acceptable pour les services réalisés est celle où nous pouvons faire un compromis pour répondre aux attentes des deux parties.

Chapter 1

Introduction

1.1 Motivations

In 2023, almost 60% of the web traffic comes from the 6 billion mobile subscribers in cellular networks all around the world [1]–[4]. This highlights the vital role of the MNOs in providing connection, services and contents to the ever-increasing users with escalating expectations.

Leading network operators adopted SDN and NFV as a solution for ultra-flexible networks that meet the diverse expectations of 5G networks. Decoupling the data and the control plane in SDN and using cloud computing technologies to virtualize, orchestrate, and scale various types of network functions to realize services in NFV, are indeed inevitable in modern networks [6].

NFV was initiated in 2012 as a result of a collaborative work between several major Telecommunications Service Providers (TSPs) [36]. By the end of the 2012, the European Telecommunications Standards Institute (ETSI) was selected as the Industry Specification Group for NFV (ETSI ISG NFV) [37]. ETSI successfully completed the first phase of its work by publishing ETSI Group Specifications [38] that included an infrastructure overview, an architectural framework, as well as MANO, security, resilience, and service quality metrics by the end of the 2013. It is worth mentioning that, being part of The 3rd Generation Partnership Project (3GPP), the ETSI specifications are accepted as standards.

The placement of the Virtualized Network Services represents one of the most crucial steps in SDN and NFV. It tackles the allocation of physical resources for network services, demanding heterogeneous resources with various Service Level Agreement (SLA) constraints. One thing that makes the placement problem interesting to research is its complexity. In [21], the placement problem is proven to be NP-Complete by reducing the bin packing problem (a well-known NP-Complete problem) to it.

The placement is encountered in many research areas and use cases in 5G networks ranging from VNF Forwarding-Graph (VNFFG) placement and Network Slicing, to virtu-

alization of the Core Network (CN), Content Delivery Network (CDN), IoT, and more [39]. Additionally, it is required for the entire lifetime of the network services, from the initial placement of the service in the substrate network, to its scaling, migration, and even its destruction.

The placement of services is carried out according to many objectives in the literature. These objectives are addressed as an optimization problem, subject to various constraints including the resources, QoS requirements, and other constraints that can be specific to the objective and the context. Resource optimizations, QoS optimizations (e.g., latency, availability, reliability, security), energy consumption, Revenue-to-Cost (R2C), and SA are among the well-studied objectives [22], [40]–[42]. We are particularly interested in SA. SA is defined as the maximum number of services that can be placed successfully over the network subject to the resources and QoS requirements.

Two general categories considered for the placement problem are *offline* and *online* placements. In an offline placement scenario, we know all the service placement requests in advance, while in an online placement scenario, we do not have any information about the future placement requests, and we need to make a placement for the current request as soon as it arrives. We may also have time restrictions for performing placements in an online placement, which makes the scalability an important requirement that needs to be considered [22]. In our thesis, we study the placement problem in an online context.

1.2 Contributions

We address some of the challenges related to the placement of the network services over the substrate networks in NFV, considering the resources and QoS requirements. Our contributions can be summarized as follows:

— *The State of the Art*: We provide a study of the related works in this domain, and we present a literature review of the placement approaches in many NFV use-cases, following different objectives and constraints. Additionally, we explore the related works addressing the placement in specific contexts like limited bandwidth in Chapter 4 and limited node resources in Chapter 5.

— *Network Service Placement with Limited Bandwidth*: Respecting the QoS requirements when there are limited network bandwidth represents one of the challenges in this area of the research. Our publications of [23]–[26] are the results of our studies in this context. Chapter 4 is dedicated to this problem, in which we cover the following

items:

- Optimization Model
- Problem Definition as a BnB Search Approach
- Applying A* Algorithm
- Node-Based vs Link-Based BnB
- Fair Distribution of the Bandwidth
- Network Defragmentation

— *Network Service Placement with Limited Node Resources*: The online placement of the network services, demanding strict E2E latency requirements, over the edge networks, with restricted computing resources presents a challenging problem which is worth investigating. Our publications of [27] is the outcome of our work in this direction. Chapter 5 is dedicated for this problem, in which we cover the following items:

- Optimization Model
- Problem Definition as a BnB Search Approach
- Applying Multiple Cost Functions and Search Strategies
- Avoiding Latency Optimization
- Fair Distribution of the Node Resources

— *Network Service Placer (NSPlacer) Tool*: In the Annex section, we demonstrate our *NSPlacer* tool, which we developed for making evaluations and comparisons between different placement algorithms including our proposed approach in [23]. In addition, it provides an infrastructure for the research community to evaluate their own placement methods.

1.3 Organization of the Manuscript

This manuscript is organized as follows. We start by introducing the background and the context in Chapter 2. Then, we traverse the state-of-the-art in Chapter 3. We study the placement of services in a limited bandwidth context in Chapter 4, and the placement of services in a limited node resources on the edge in Chapter 5. We conclude in Chapter 6. The representation of our *NSPlacer* tool is located in the Annex section.

Chapter 2

Background

2.1 5G

The continuous evolution of telecommunication technologies have led to higher performance expectations. While it may seem logical to expect 10x better performance from 5G compared to 4G, similar to the increase seen from 3G to 4G, in reality each technology has limits. To meet these expectations, incremental improvements may not be enough. Instead, 5G is embracing more promising innovations, departing from previous technologies. These performance expectations and requirements are driven by various pioneering use cases needing either greater bandwidth, improved quality of service, or more supported connections. These requirements are illustrated in three main categories, in Fig.2.1, according to the ITU [5], which also includes some related use-cases:

— *eMBB*: eMBB concerns in providing the bandwidth which is required by several use-cases that demand streaming UHD 3D videos like in AR. Additionally, replacing the wired connections by their wireless alternatives capable of providing at least the same amount of the bandwidth, for many different use-cases ranging from 5G Verticals via Ethernet copper connections, or home internet fiber connections, or even backhole connections between the core network and the edge networks, is another ambition which is followed in this category. The mentioned need stems from the annoying cost of the establishment and the maintenance of the wired connections.

— *mMTC*: mMTC regards to grant the coverage of a massive number of connected objects in Smart Cities and IoT systems, from the Smart Grids in the energy sector to the sensors/actuators in the agriculture, the water utilities, health care, etc.

— *URLLC*: URLLC focuses on delivering the strict latency and reliability requirements, which is encountered in use-cases like Smart and Autonomous Vehicles, Intelligent Transportation Systems, Remote Surgery, etc.

Note that a specific use-case represented in one category like Autonomous Vehicles

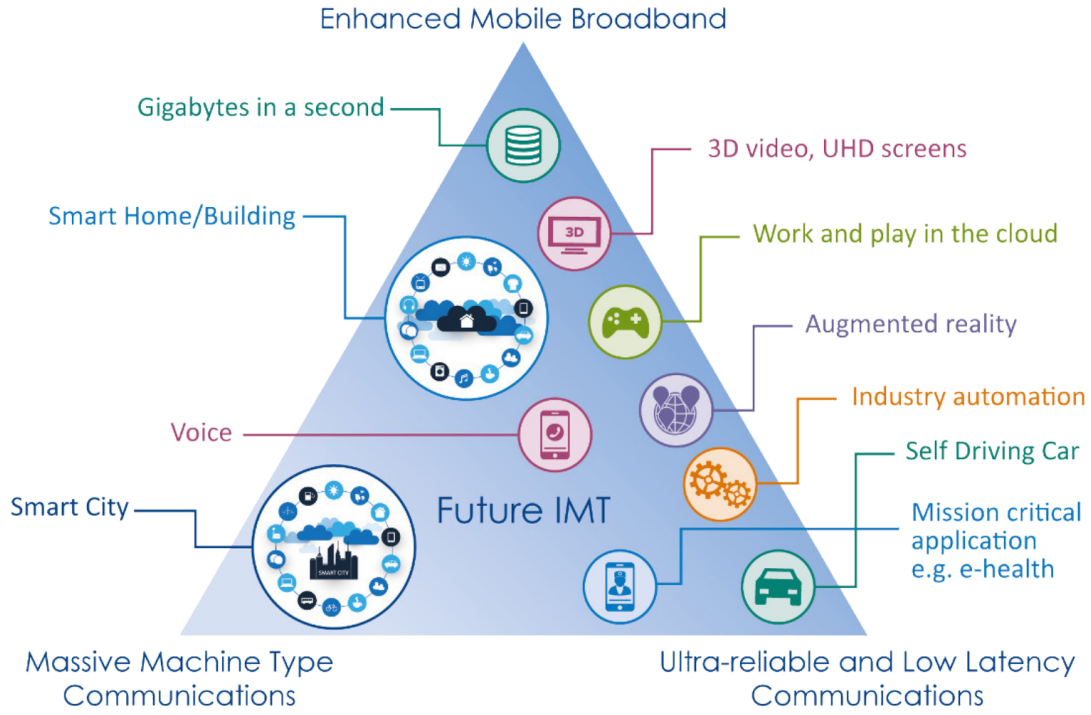


Figure 2.1 – 5G use-case categories [5]

in URLLC, alongside demanding a strict latency and reliability, it has bandwidth and connectivity requirements, which are the primary topics in eMBB and mMTC respectively. In fact, a use-case is identified in a category, when it touches the limits of the technology in that specific category (*i.e.,* its other requirements are feasible to be met by the current technology).

The ambitious expectations from the fifth-generation networks can not be met simply by improving the technology of the current and the previous generation networks. To conform to these requirements an on-going revolution started over the last decade on the telecommunication technologies. Several new paradigms have emerged like SDN and NFV, which demonstrated a high potential in creating flexible and scalable modern networks.

We have generally two types of operations to realize a network: packets' transmission and packets' treatment. To introduce SDN and NFV in a nutshell, SDN has emerged as a solution to facilitate the packet transmission, while NFV as a remedy to help with the packet treatment. Note that the whole story is intended to get rid of the monolithic black-box vendor-specific solutions that were almost impossible to maintain and update in order to meet the constantly-changing telecommunication requirements. The need to decompose the complex network operations and to have a standard and open solutions,

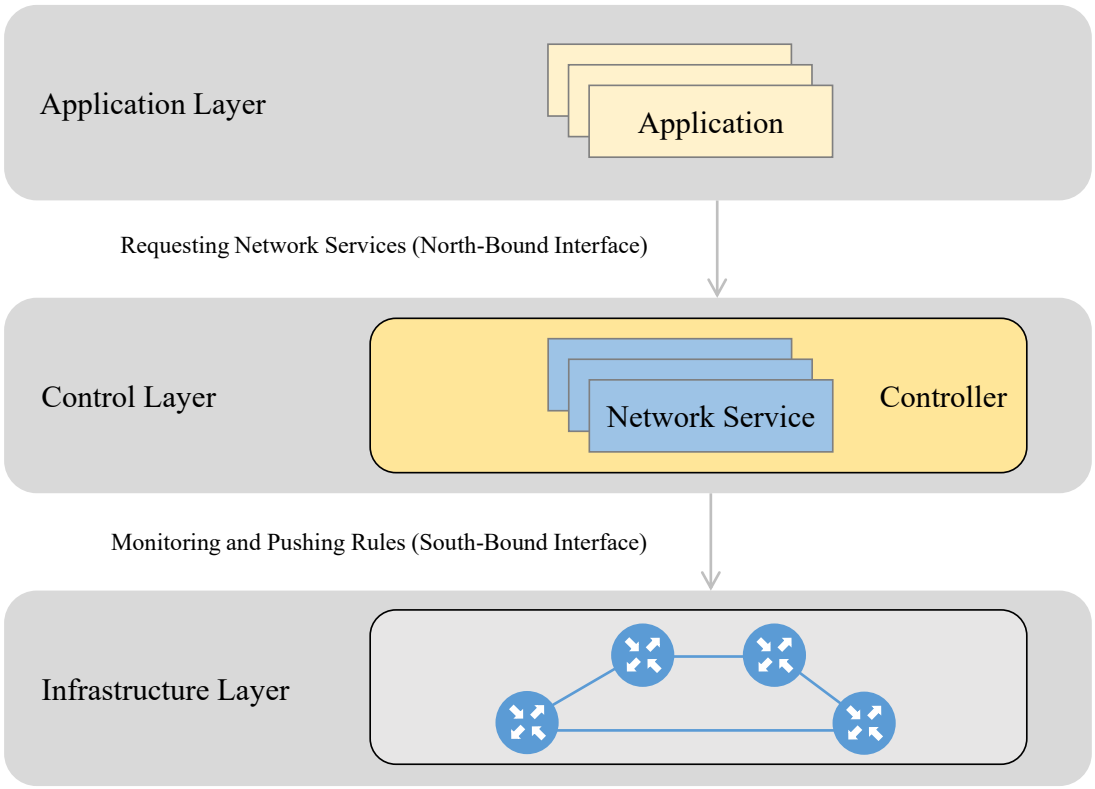


Figure 2.2 – SDN Architecture

drives the interests in SDN and NFV.

2.1.1 SDN

Previously, the traditional routers were responsible to decide the routing of the packets in a distributed manner. This decision was made with the help of a routing table within a router, and each router is responsible to create and update this table by constantly discovering and monitoring the topology of the network, and the capacity of each network link based on its actual traffic load. The complexity of the synchronization between these distributed routers in a continuously changing traffic load on one hand, and having different vendor-specific routers with proprietary routing algorithms on the other hand, creates big obstacles for the packet transmission to meet with the strict QoS requirements.

SDN [7] has emerged as a remedy to reduce the mentioned complexity by centralizing the control of packet transmission. SDN replaces routers by simple switches, and a centralized controller is responsible for monitoring the network and updating the routing tables of the switches in an open and standard way.

The concept of the centralization of the control of the packet flows dates back to the very first beginning of the Internet [8]. Formerly, the scalability was by far more important than recently emerging QoS requirements like we mentioned earlier in Section 2.1. Consequently a distributed approach was preferred against a centralized one in the past. Today, everything has changed. Achievements in cloud computing, Artificial Intelligence (AI), and open-source community have changed many areas, and telecommunication is not an exception, up to the point where a centralized control now seems more suitable.

A detailed architecture can be found in [7], but ONF represents a more concrete architecture of SDN with its three main layers in [9] (shown in Fig. 2.2). The application layer requests for the required network services through a north-bound Application Programming Interface (API) to the SDN controller, and the controller monitors and pushes the rules to the network switches via a south-bound API. The infrastructure layer or data layer is comprised of the switches which are responsible to forward the packets through the network.

2.1.2 NFV

NFV has been appeared to softwarize and virtualize the network functions, which were previously provided by dedicated hardware devices, to enable their execution in COTS industry-standard high-volume servers [10]. The main obstacles particularly related to maintaining, updating, scaling and migration of the PNFs on the one hand, and the advancements in the virtualization technologies and the cloud computing on the other hand pushed the telecommunications ecosystem to move toward NFV without hesitation. NFV, not only overcomes the mentioned barriers, but also it reduces the costs by sharing the resources, and it leads us toward the dream of automation and zero-touch networks.

Although NFV brings many advantages, it presents several unprecedented challenges. First, the new softwarized VNFs need to be at least as performant, reliable and secure as their hardware equivalents. We need to take into account that the transition would take time, so the compatibility and the coexistence with dedicated PNFs is inevitable. NFV comes with the functions which are responsible for the management, the orchestration, and ultimately the automation of the VNFs. The integration, the interoperability, and having the standard APIs between the orchestrator and the VNFs also need to be addressed.

A network service traditionally comprised of a set of PNFs (*e.g.,* routers, firewalls, load balancers, etc.) linked together to provide a network utility. With the advent of the NFV, a network service is represented as a logical graph of VNFs lined together via the

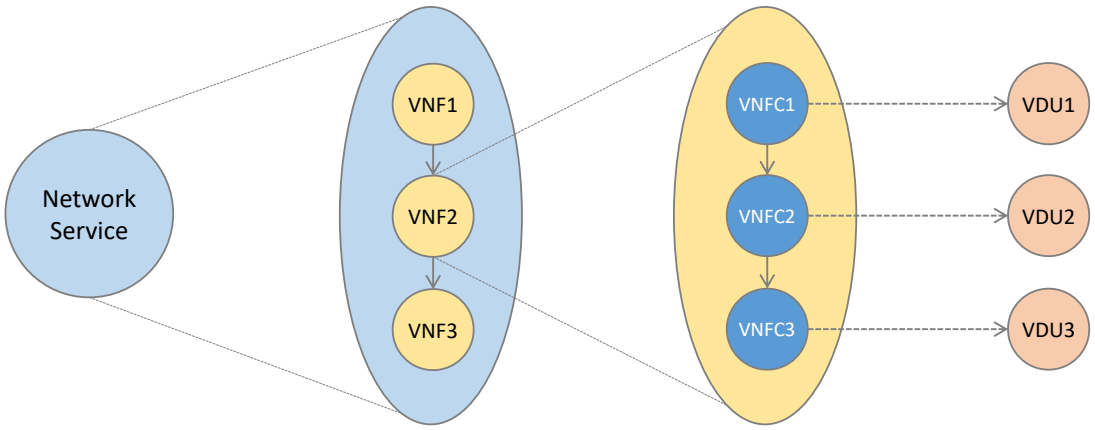


Figure 2.3 – Network Service Decomposition

VLs according to specifications made by ETSI [11]. A VNF is considered as the smallest component from the network viewpoint, and it can itself be decomposed as a graph of VNFCs that are connected with VLs. A VNFC is considered as the smallest component from the resource viewpoint, which is deployed over a VDUs. The Fig. 2.3 demonstrates the mentioned decomposition.

The implementation of the proposed network service decomposition is out of the scope of ETSI specifications, but to mention some of the the most popular technologies that are frequently chosen for developing, we can select MVC as a software design pattern [12], Microservices as a software architecture [13], Docker as a container manager [14] and Kubernetes as a container orchestration system [15].

2.1.3 SDN/NFV Enablers and Solutions

To realize the proposed SDN and NFV architectures, many enablers exist like CC, MEC, Serverless Computing (SC), container orchestration tools, etc., as well as frameworks and specifications like ETSI MANO, SFC, Network Slicing, etc.

Furthermore, many solutions are proposed and supported by the open-source community, leading network operators and network equipment providers. Some of them are listed as following:

— *Open Source MANO (OSM)* is an ETSI-hosted project to develop an Open Source NFV Management and Orchestration (MANO) software stack aligned with ETSI NFV [16].

— *ONOS* is proposed by the Linux Foundation as an SDN controller in the form

of a scalable high-performance high-available open-source distributed Operating System (OS) [43]. Note that, a high-available distributed OS for a SDN controller may sound in contrast with the concept of the centralization in SDN, but we should be aware that we have logically one SDN controller and its several synchronized replicas for the availability reasons.

— *Open Network Automation Platform (ONAP)* is an open-source orchestration and automation framework hosted by the Linux Foundation [17].

— *OpenDaylight* is a project which was started as a concept demonstration and a prototype of SDN and NFV, and then it was proposed as a product for the monitoring, automation, and scaling of the networks with different sizes with the help and support of the Linux foundation, Red-Hat, Orange, Ericsson, Cisco, etc. [18]

— *OpenAirInterface (OAI)* is an open experimentation and prototyping platform created by the Mobile Communications Department at EURECOM to enable innovation in the area of mobile/wireless networking and communications [19].

— *OpenFlow* is chosen as an standard API to connect the SDN controller to the network switches [20].

— *Open vSwitch (OVS)* is an open-source implementation of a distributed virtual switch fully compatible with the OpenFlow API.

2.2 Summary

In this chapter, we introduced the general context of thesis, which is the 5G networks, the use-cases and the enablers especially SDN and NFV. We also listed some of the current prototypes and solutions that are proposed and used in the industry and by the research community.

Chapter 3

State of the Art

3.1 Introduction

In this chapter, we review the related works that study the problem of the placement of VNFs and Cloud-native Network Functions (CNFs) in 5G networks.

One of the most important challenges encountered in NFV and SDN, is the placement of the VNFs, and routing the traffic between these VNFs (*a.k.a.* VL placement). A placement algorithm determines where to place the VNFs considering the resources and the QoS requirements. It is confronted frequently from the initial placement of the VNFs to their scaling, their migration, and even their destruction. This is the main interest and the concern on the current work.

In the context of SDN and NFV, a Network Service (*a.k.a.* Service Function) is defined as a set of VNFs (or CNFs) and VLs creating a graph of the network service (*a.k.a.* SFC). VNFs demand for a subset of computing and storage resources provided by the substrate network nodes, and the VLs specify the data flow and traffic between these VNFs. Placing the VNFs concerns assigning them to the corresponding network nodes, and the VLs are realized by routing the paths between the placed VNFs. The rule of thumb is that when there is virtualization of the network services and the allocation of the resources, we talk about NFV, and who says the network links and the routing, says SDN. The placement of the VNFs and VLs is defined in the context of both NFV and SDN, since placing the VNFs impacts the routing to a great extent and vice versa. Thus, they need to be considered simultaneously [44]–[47].

The surveys are the valuable sources of information helping us to explore a research domain more efficiently, and hopefully we can find several interesting surveys in our dynamic area of research which is the VNF/CNF placement. Among this surveys we can mention [48]–[52].

3.2 Algorithms

There are several categories of the algorithms that are used for the placement problem. We will mention some of them in the following.

3.2.1 Exact Methods

Exact methods are a set of algorithms that guarantee to find optimal results by exploring the possible solutions based on a mathematically model, which defines the objectives and constraints (*e.g.,* ILP, MILP, Column Generation (CG), etc.) [53]–[56].

3.2.2 Heuristic Methods

Heuristic methods are problem specific solutions, that are relatively fast and scalable, but are not necessarily of high quality and optimal (*e.g.,* Random-Fit (RF), First-Fit (FF), Best-Fit (BF), etc.) [57], [58].

3.2.3 Meta-Heuristic Methods

Meta-Heuristic methods are considered as an extension to heuristic methods, which are problem-independent algorithm design patterns which can achieve near-optimal results by iteratively optimizing the results based on an evaluation metric (*e.g.,* Genetic Algorithm (GA) [59], Particle swarm optimization (PSO) [60], Ant Colony Optimization (ACO) [61], [62], and Tabu Search (TS)).

3.2.4 Machine Learning Based Methods

Machine Learning (ML) methods are a subset of AI algorithms that can learn from a data or a pattern and reproduce the results for the new data. They can be categorized into several big classes of the algorithms including supervised learning, unsupervised learning and Reinforcement Learning (RL). The latest techniques in RL for the placement problem have attracted many researchers recently [31], [63]–[65]

3.3 Constraints and Objectives

There are many constraints related to the placement problem that are addressed in the literature. A constraint can be treated in two different ways. It can be seen as a constraint that needs to be satisfied (for example by considering a threshold), or it can be seen as an objective of an optimization problem to be minimized or maximized. One study may considered several constraints and it may choose a subset of these constraints as its objectives (*a.k.a.* multi-objective). Some of the most frequently encountered constraints are mentioned as follows.

3.3.1 Cost

Cost usually represents the monetary cost of using a cloud or network resource, but it can also be interpreted as a cost function in an optimization problem.

In [66], the authors try to optimize the operational and network traffic cost of the placement by proposing an algorithm based on markov approximation called SAMA, by considering energy consumption.

In [67], the authors try to address the placement of the VNFs by minimizing the cost of the resources in MEC. First, they propose an MILP optimization, and then they propose a GA approach for large networks. The obtained results confirm its superiority in comparison with the benchmark heuristics like RF and FF.

In [53], a cost-optimization scalable ML-based placement algorithm is proposed by considering cost of the resources and latency. First they propose an ILP model, then they propose their algorithm called MAPLE for large scale networks. They divide the network into distinct clusters using k-medoids clustering method to reduce the complexity of the placement.

3.3.2 Cloud Resources

Cloud resources represent CPU, RAM, storage, General-Purpose Graphics Processing Unit (GPGPU), Field Programmable Gate Arrays (FPGA), Digital Signal Processing (DSP), and any resources that can be provided by a network node to the VNFs or CNFs.

Although optimizing of the resource allocation relies on the premise that the amount of the required computing resources by the VNFs and the required bandwidth by the VLs are known in advance, in reality it is not very clear. Additionally, the dynamic nature of the

network load directly influences these requirements. We could perform optimal placements, but we always have to accept a risk of performance degradation which may arise because of under-provisioning, or resource waste due to over-provisioning.

In [68], the authors try to minimize the amount of the provisioned computing resources by regulating the required E2E latency. They consider the processing delay of a packet by the VNFs with some parameters in their model. By knowing the minimum delay of processing a packet by a VNF while allocating the maximum computing resources for it, and the maximum delay while allocating the minimum resources, they optimize the provisioned resources while respecting the E2E latency by using Mixed Integer Quadratically Constrained Programming (MIQCP).

In [69], the authors proposes a resource allocation optimization framework, and a real-time provisioning and auto-scaling algorithms in network slicing according to cloud-native architecture.

3.3.3 Network Traffic

In [70], the authors investigate the practical feasibility of MEC by proposing a Chain-based Low latency VNF ImplemeNtation (CALVIN) framework. They propose a distributed mapping with one VNF per Virtual Machine (VM) in a MEC, and they modified the Linux kernel to reduce the service latency by avoiding the metadata structure processing and batch processing of the packets which occur in the conventional Linux networking stack.

One of the interesting topics in the 5G networks is the muticast traffic routing. [44] proposes a cost-optimization MILP placement solution for VNFs and VLs with multipath routing in 5G CNs. They consider the latency requirements in multicast routing, and they propose a heuristic algorithm for large scale networks.

In [57], the authors propose an ILP optimization and a heuristic to minimize E2E latency in multi-tenant environment.

3.3.4 Revenue

Revenue represents the monetary profit of placing the network services usually for the network or service provider [54], [71].

[54] proposes an MILP approach for 5G vertical services to maximize the mobile network operator profit, and a meta-heuristic algorithm called MaxSR for large scale networks.

3.3.5 Acceptance Ratio

Acceptance ratio represents the maximum number of network services that can be placed over the network considering the same amount of resources. It is usually motivated by the the monetary profit of placing network services [31], [56], [71].

3.3.6 Energy Consumption

Energy consumption represents the amount of the energy consumed whether in big Data Centers (DCs) or small IoT devices, for economic and/or ecologic reasons.

One of the solutions to reduce the energy consumption is to place as many VNF as possible over the minimum number of network nodes (*a.k.a.* VNF consolidation) to turn-off the rest of the network. Although, this approach can save some energy, it leads to hardware competition and performance interference. To address the mentioned problem, [72] proposes a model to make a compromise between the energy consumption and the performance. Then, it tries to solve it by DRL and a heuristic. [73] is another similar paper that tries to optimize the energy consumption with DRL.

3.3.7 Scalability

Scalability is considered as one of the most important QoS criteria, which represents to what extend an algorithm can be scaled in size without significant degradation in execution time or quality.

In [31], the authors propose a DRL placement approach to optimize the allocated resources considering the QoS requirements. For the large scale networks, they come up with modifying the basic Deep Deterministic Policy Gradient (DDPG) by a Heuristic Fitting Algorithm (HFA) which makes an estimation of the best allocation policy to boost the convergence of the DDPG (they call it E^2D^2PG). Besides, they demonstrate the scalability of E^2D^2PG for large networks in their evaluations. In [71], the authors proposes a scalable placement algorithm based on DRL with Graph Neural Networks (GNNs) to maximize the acceptance ratio and the average revenue.

3.3.8 Throughput

Throughput represents the number of packets or tasks that are transmitted or performed successfully in the network.

In [55], the authors propose an ILP optimization and a heuristic to maximize the network throughput in MEC and smart cities considering different QoS requirements.

[62] considers multiple constraints in placing the network services like resource utilization, energy efficiency and throughput.

3.3.9 Availability

Availability is considered as one of the most important QoS criteria, which represents a quality in which a service is always ready to carry out its tasks when it is needed.

Considering the redundancy to increase the availability of the critical stateful VNFs, we have an active VNF, several stand-by VNFs. The active VNF transfers its state periodically to the stand-by VNFs in order to migrate in case of a failure. [74] proposes a RL placement approach by minimizing the cost of the resource allocation, as well as the cost of a continuous state transfer between active and stand-by instances, considering the required latency, and service acceptance ratio. Another work which considers the placement of the stateful VNFs is [56]. They propose, first, an ILP cost-optimization solution, and then a heuristic method. They make a real test-bed for their experimentation, and they showed that their algorithm increases the fault tolerance and acceptance ratio.

3.3.10 Reliability and Fault Tolerance

Reliability and fault tolerance are QoS criteria, which represent the robustness of a service in case of an undesirable incident.

In [58], the authors try to propose a heuristic cost-optimization placement approach based on bin packing problem for URLLC services. In addition to the latency, they increase the reliability by taking into account the user mobility, and reducing the number of handovers. Following the strict reliability constraints, they even consider the battery limitations of IoT devices.

3.3.11 User-location

User-location is considered on the edge, and it effects the placement of the services. In addition, user-mobility may trigger the service migration which may require a re-placement of the service.

In [75], authors propose a cost-optimization MILP-based approach in MEC, by minimizing the service provisioning cost and transport network usage, considering user-location

and latency. In addition, they propose a heuristic algorithm to tackle the scalability of their MILP-based algorithm.

In [76], the authors propose a VNF placement and routing ILP optimization algorithm to minimize total resource consumption in network slicing by considering the user location and E2E latency.

3.3.12 Performance

Performance represents the quality to determine how well a network service accomplishes its tasks usually in terms of the response time [72], [77].

3.3.13 Security

Security represents the quality to determine how protected a network service is from threats, attacks, risks, and potential harm.

One of the problems that arise by sharing a host between several VNFs is that, if a host undergoes an attack, all of its related VNFs are effected. [78] tries to increase the revenue and decrease the cost by increasing the security of the placing network service by adding security VNFs (*e.g.,* firewalls, Deep Packet Inspection (DPI), Intrusion Prevention System (IPS), etc.).

3.4 Use-Cases

The placement problem is confronted in different network domains (Core/Transport/Access networks, or Cloud/Fog/Edge from the NFV viewpoint) and different 5G use-cases. They can generally be classified into three main categories of mMTC, URLLC and eMBB, but to be more concrete, we can mention smart cities [55], verticals[54], IoT, 4K 8K video streaming [79], CN virtualization [80], Cloud Radio Access Network (CRAN) [81], [82], Autonomous Vehicles (AV), etc.

In [83], the authors study the problem of optimal placement of RAN services in network slicing context by MIQCP for minimizing the latency and power link capacity, and maximizing the data throughput for reserving the bandwidth of the links for eMBB services.

3.5 Enablers

3.5.1 Network Slicing

Network slicing is an interesting subject in 5G networks. In [84], the authors study the management and the orchestration of the network slices in 5G, fog, edge and clouds. Network slicing and the placement problem are closely interconnected. Network slicing is encountered in different layers of the 5G networks, from the physical layer by allocating the radio spectrum in Radio Access Network (RAN) [85] to the application layer. In the application layer, the network slices are realized by the network services where we encounter the cloud resource provisioning for the SFCs, the same as the VNFFG placement [86], [87].

In [79], the authors propose a heuristic placement algorithm called Adaptive Interference Aware (AIA) in network slicing for placing IoT services (like autonomous cars or 4K/8K video streaming) to maximize the throughput of accepted requests.

3.5.2 Automated Orchestration

One of the other interesting subjects in 5G networks is automated orchestration. Not only the orchestrator decides where to place the network services, but also it determines when to scale or migrate the already placed services. In [88], the authors propose an automated spatial resource allocation approach based on RL, to automatically determine the migration time between DC and MEC by considering the resources and E2E latency.

3.5.3 Multi-access Edge Computing (MEC)

MEC is one of the enablers of URLLC services. [89] proposes a placement algorithm for the URLLC services in MEC with a multi-objective model by optimizing the latency and availability.

In [90], the authors try to optimize the energy consumption in MEC by using Constraint Programming (CP) and taking into account latency in muti-tenant networks.

3.5.4 CNF Placement

One of the most popular subjects that we frequently encounter in the literature recently, is related to CNF placement.

What makes difference between a PNF and a VNF is the fact that a VNF is a softwarized version of a physical dedicated network function. Although the mentioned definition does not talk about how to execute (deploy) a VNF (whether over a Virtual Machine (VM), a container, or even a machine without any virtualization), to distinguish a VNF from CNF, we suppose that a VNF is deployed over a VM and a CNF over a container.

Although, they may sound similar, they are two different technologies that need to be considered for the placement. From the technical point of the view, a container is an isolated process in the host OS that has limited access to the resources (from CPU, RAM, to file system, network interface, etc.). Sharing an OS, reduces the overhead of the VMs and it brings improved performance, increased scalability (allowing to have more containers than VMs considering the same hardware resources), and better manageability. [91] makes an interesting comparative study between containers vs VMs.

Even though, containers bring many advantages, by sharing a common OS, they may fight for the shared resources, which may lead to performance interference. To provide a guaranteed performance in this context is considered as one of the biggest challenges which should be considered in container placement algorithms (*a.k.a.* container scheduling algorithms). [92] proposes algorithms for container placement and reassignment to address the mentioned performance degradation problem.

In [93], a literature review is made over the current container placement models in edge computing context.

In [60], the authors propose an ILP optimization, PSO and greedy algorithms for container placement of IoT services in dynamic fog computing.

[94] proposes a CNF placement algorithm to minimize the resource utilization by considering the availability.

[59] addresses the container placement problem by proposing Improved Genetic Algorithm (IGA) to minimize the energy consumption. Among the other works that uses GA, we can mention [95], which tries to find near optimal container placements by considering the inter-container traffic and application response time.

[96] tries to minimize the energy consumption and resource utilization by Whale Optimization Algorithm (WOA).

One of the challenges in this area is the placement of the entire cluster of containers considering the inter-container traffic. [97] addresses the online container cluster provisioning by considering the inter-container traffic.

Similar to the VNF placement, the security challenges are encountered in the container

placement problem. In [77] the authors propose a GA container placement algorithm to address the co-resident attacks, considering the performance interference. To reduce the convergence time of their proposed GA, they use a Simulated Annealing (SA) algorithm.

Recently, serverless computing is gaining popularity for its capability to reduce the complexity of the management of the provisioning and the scaling. [98] proposes optimization method to minimize task execution time, up-link usage, and cloud execution cost for container scheduling problem in serverless edge computing, considering the resources, traffic and proximity for the data-intensive services. They also implement a prototype that targets the container orchestration system Kubernetes.

3.6 Summary

In this chapter, we tried to explore the literature of the placement problem by considering different approaches. From different algorithms that are applied to this problem to its constraints, objectives and enablers. Note that a work usually covers many aspects and a work that is categorized in the section of *e.g.,* the energy consumption, it could have been categorized in other sections *e.g.,* E2E latency, or performance. We tried to spread the works by trying to make sure that the selected work can represent well its category and each category has at least one sample.

Chapter 4

Network Service Placement with Limited Bandwidth

4.1 Introduction

In this chapter, we address the problem of the online placement of services in a limited bandwidth context. We demonstrate how to obtain optimal results (the distinguishing advantage of ILP-based placement methods) while maintaining the scalability of a heuristic-grade placement strategy.

First we define the mathematical model of our problem in ILP to be able to address it in an offline placement scenario, and to obtain the global optimal results. Then, we formalize it in CG, to improve its scalability.

Next, in Section 4.4, we propose our BnB-based approach. BnB caught our attention for several reasons. Having a large number of constraints inherent to the placement problem can be beneficial to BnB (the more constraints we have, the more we can prune the tree, which leads to a faster search procedure). Besides, it allows us to apply different sophisticated AI search strategies (especially A*) to obtain optimal results. Moreover, it carries an intrinsic potential for parallel execution. Therefore, we believe that BnB can be a suitable solution for online placement.

This chapter is organized as follows. First, we explore the related works in Section 4.2. We present our formulation of the problem by an ILP and CG approach in Section 4.3. Next, we describe in detail our BnB approach in Section 4.4. Then, in Section 4.5 we plunge into the evaluations to investigate the effectiveness of our solution. Finally, we conclude in Section 4.6.

4.2 Related Work

A significant part of the placement algorithms involves the exact resolutions, formulating the problem as ILP or MILP, and trying to solve it via solvers and optimizers.

In [28], the authors aim to optimise the placement of multi-constrained services, by considering the bandwidth and the latency for the links, and different system resources for the nodes on an edge-core star-based network topology graph. Their objective is to maximise the number of placed service requests, first by proposing an ILP resolution, and then a heuristic. In an effort to ensure a fair distribution of the resources, they considered the cost of using a resource proportionate to the remaining capacity of the corresponding node or link (*i.e.,* the cost of using a resource on an overloaded node/link is much more than an underloaded node/link). The researchers then demonstrated that this slight modification results in a more fair distribution of resources and leads to an increase in the number of accepted services.

Despite the optimality and the clarity of ILP-based methods, they suffer from scalability issues. Note that performing an exact resolution requires a long processing time until obtaining an optimal solution, and it can only be used for small network topologies. This limitation presents a significant barrier in online placement scenarios with real-time constraints.

When it comes to the scalability of approaches, heuristics are often preferred. Best-Fit or Decreasing First-Fit is one of the most well-known and efficient placement heuristic, which sorts the nodes and places the VNF over the node containing the maximum amount of available resources iteratively. [29] explores scalable and cost-efficient heuristic algorithms, especially the Best-Fit algorithm, and it proposes a multi-stage graph algorithm to find near-optimal solutions in a scalable manner. They assumed that service requests arrive randomly, considering a random life span for each service. The services are comprised of 3 VNFs, requiring random units of resources for VNFs and VLs. The simulation duration is fixed to a time unit for each run, and they investigate the execution time (including the scalability of the Best-Fit and the Multi-Stage algorithms), acceptance ratio, and average cost of the placement in terms of the assigned resources.

Although, designed to obtain a solution in a reasonable time frame, heuristics do not provide quality guarantees, and the risk of getting stuck in local optimums is extremely high.

Meta-heuristic and evolutionary algorithms are another direction addressing the place-

ment problem. In [30], a genetic algorithm based approach is proposed. They compared their results against the Best-Fit algorithm as a baseline algorithm, and they achieved higher request acceptance rates, stemming from more efficient resource utilization compared to a baseline greedy algorithm with a similar optimization objective.

Despite being powerful in escaping from local optimums, they suffer from unpredictability, especially in terms of execution time, which is extremely important in online placement.

Advanced AI search mechanisms and recent developments in machine learning techniques, and particularly DRL, have gained lots of attention due to their promising potential.

The authors in [31] explore the potential of DRL for maximizing the number of accepted services, satisfying the QoS requirements. They show that their DRL approach can learn the non-linear relation between QoS metrics and traffic, in order to be able to find better placements, resulting in more accepted services. [32] tries DRL with RGCN for maximizing the service acceptance. In [33], a DRL approach utilizing Double Deep Q Networks is proposed to reduce the rejection ratio of the service requests, while improving the system's scalability through an offline training. A Q-Learning approach is investigated in [34], trying to establish a compromise between scalability and optimality in order to maximize the benefit of the service provider. They could obtain near-optimal solutions in a relatively short time, while considering the network load balance.

Our work falls into the latter category, discovering the potential of AI search strategies, in particular A^*. It is worth specifying the position of our work among ML-based approaches. In our contribution, we are able to find optimal (in the worst case near-optimal) placements, while aiming to be as scalable as possible. Whereas, in ML-based approaches, these methods are computationally efficient after the training phase, with the target being to obtain placements as close as possible to the optimal and near-optimal results by exploiting novel learning approaches. These are the important differences between our work and the recent ML-based approaches.

4.3 Model

In this section, we define our placement problem model, and then we try to resolve this model by ILP and CG. The obtained results are presented later on the experimentation section.

4.3.1 The problem definition

The service placement problem consists in placing a service in the form of a graph on a substrate network, which is also described by a graph. Thus, a service graph $k \in K$, where K is the set of services, is denoted by a directed graph $H_k = (F_k, L_k)$, composed of a set of VNFs F_k, connected via a set of VLs L_k, accompanied with SLA, reflecting the QoS requirements. A VNF $f \in F_k$ requires a subset of resources, here we consider CPU C_f, and each VL $l \in L_k$ connecting two VNFs requires an amount of bandwidth B_l. Similarly, a Substrate Network (SN) is a physical network represented by a directed graph $G(V, E)$ with its corresponding nodes V and links E. A node $v \in V$ has a capacity of resources, here we consider CPU denoted by C_v, and a link $e \in E$ has a bandwidth capacity denoted by B_e, as well as QoS metrics (here we consider latency).

The placement problem deals with finding mappings of a service's VNFs and VLs into network nodes and network paths, respectively. The notations of the model are summarized in Table 4.1.

Table 4.1 – Notations

Name	Description
$G = (V, E)$	Substrate network
V	Set of nodes of the network
E	Set of links of the network
$\omega^+(v)$	Outgoing neighboring nodes of $v \in V$
$\omega^-(v)$	Ingoing neighboring nodes of $v \in V$
B_e	Bandwidth capacity on the directed link from $v \in F_k$ to $j \in w^+(v)$
$H_k = (F_k, L_k)$	Virtual graph of service k
F_k	Set of VNFs to be placed for service k
L_k	Set of VLs between the VNFs of service k
$\omega^+(f)$	Outgoing neighboring VNFs of $f \in F_k$
$\omega^-(f)$	Ingoing neighboring VNFs of $f \in F_k$
B_l	Bandwidth requested between the two VNFs of $l \in L_k$
C_f	Computing resources requested by $f \in F_k$
C_v	Computing resources capacity at $v \in V$
D_e	Delay experienced on link $e \in E$
D_k	Delay requested for service $k \in K$
D_l	Delay requested between the two VNFs of $l \in L_k$

4.3.2 Compact formulation

Variables The ILP contains two sets of binary variables: the first set, x, represents the routing decision, and the second set, y, the service placement decision. More precisely, the variable $x_e^{l,k} = 1$ if the virtual link l of service $k \in K$ is using the link e with $e \in E$; and 0, otherwise. The variables $y_v^{f,k} = 1$ if the VNF $f \in F_k$ of the service $k \in K$ is instantiated in node $v \in V$; and 0, otherwise.

Constraints We consider the CPU as a node resource, and the bandwidth as a link resource. In addition, in terms of the QoS metrics, latency is considered for the links and E2E latency for the entire service. Note that, our approach does not impose any limitations for adding more resource types or more QoS metrics for nodes, links, or services. A VNF can demand a set of resources of different types (*e.g.,* CPU, GPU, RAM, storage, FPGA, etc.), which can be provided by several network nodes at the same time. Consequently, we consider two sets of capacity constraints:

$$\sum_{k \in K} \sum_{l \in L_k} B_l x_e^{l,k} \leq B_e, \tag{4.1a}$$

which is the bandwidth capacity constraints of each link $e \in E$; and

$$\sum_{k \in K} \sum_{f \in F_k} C_f y_v^{f,k} \leq C_v, \tag{4.1b}$$

which is the CPU capacity constraint of each node $v \in V$.

First, we ensure that the VNFs of a service are only deployed once throughout the network with the following constraints:

$$\sum_{v \in V} y_v^{f,k} \leq 1 \qquad \forall k \in K, \forall f \in F_k. \tag{4.1c}$$

Moreover, VNFs of a service $k \in K$ cannot be deployed on the same network node $v \in V$ (*i.e.,* the VNFs of a service are placed on separate network nodes), which is ensured by:

$$\sum_{f \in F_k} y_v^{f,k} \leq 1, \qquad \forall v \in V, \forall k \in K. \tag{4.1d}$$

For each service $k \in K$, we ensure the virtual link $l = (f_i, f_j) \in L_k$ flow conservation

at each node $v \in V$ with:

$$\sum_{e \in \omega^+(v)} x_e^{l,k} - \sum_{e \in \omega^-(v)} x_e^{l,k} = y_v^{f_j,k} - y_v^{f_i,k} \tag{4.1e}$$

Finally, we bound the E2E service latency. For the sake of simplicity, we consider the sum of the latency required by VLs as the E2E service latency, and the sum of the latency provided by placed network paths with the following constraints:

$$\sum_{e \in E} D_e \sum_{l \in L_k} x_e^{lk} \leq D^k \qquad \forall k \in K, \tag{4.1f}$$

$$\sum_{e \in E} D_e x_e^{lk} \leq D_l \qquad \forall k \in K, \forall l \in L_k. \tag{4.1g}$$

Objective The objective function is defined as:

$$\max \sum_{k \in K} \sum_{v \in V} y_v^{f_0,k}, \tag{4.1h}$$

where f_0 is the "first" function of the service. We only need to count the number of "first" functions in the objective since the flow conservation constraints ensure that a solution contains no partial embedding.

Cuts An ILP solver's performance depends on the problem's relaxation, *i.e.,* the problem without integrality constraints. It evaluates the nodes of its Branch and Bound tree with the linear relaxation of the problem. If the linear relaxation solution at the current node is worse than the best integer solution found so far, it can prune the node. The linear relaxation quality (the gap between the integer and relaxation solution) significantly impacts the resolution time since a good relaxation will lead to a faster exploration of the tree. We can improve the lower bound by devising cuts. These constraints are not necessary for the correctness of the model as they are implicitly satisfied by the constraints of the base integer model, but they are usually violated in the relaxed version of the problem.

This formulation has a poor relaxation, however, we can improve it by exploiting the one VNF per node constraint. Since nodes can host only one VNF of the same service, we can define the minimum amount of bandwidth used by each service. For example, let's consider a daisy chain service graph with 3 VNFs (F1, F2, and F3), and 4 VLs connecting F1 to F2, F2 to F1, F2 to F3, and F3 to F2. If each VL would require 1 unit of bandwidth, at least, we require four units of network bandwidth. In mathematical form, we can express the minimum bandwidth usage as:

$$\sum_{k \in K} \sum_{v \in V} \sum_{l \in L_k} B^{l,k} y_v^{f_0,k} \leq \sum_{e \in E} B_e. \tag{4.1i}$$

4.3.3 Embedding Decomposition

The previous formulation lacks scalability. To solve larger instances, we need to use decomposition methods. These methods are popular tools to make scalable ILP-based algorithms and they also provide higher bounds to evaluate the solutions.

Multiple decompositions are available. We focus on the *embedding decomposition*, where each variable represents a possible embedding of the service onto the physical network. This decomposition shines when multiple services share the same configuration (same bandwidth, same number of CPUs, and same latency requirements) as they can share the same embedding and we can easily compute the resource allocation.

Since the number of embeddings is exponential, an embedding-based formulation would require an exponential number of variables (columns). However, a solution only contains a few of them. To generate only useful columns, we use the CG algorithm [99]. The algorithm solves large-scale linear programs via a back-and-forth resolution of a *master problem* and one or multiple *pricing problems*. Starting from a *reduced* master problem, the master problem feeds dual values to the pricing problems, and the pricing problems feed improving columns to the master problem. For each service k, we need to generate an embedding γ among all possible embeddings Γ_k for the service k.

Master Problem

We only need the set of variables $z_{k\gamma} \in \mathbb{N}$ to indicate the number of services k allocated on the embedding γ. Each embedding γ is defined by the amount of bandwidth it uses on each link e, denoted by $\delta_e(\gamma)$, and the number of CPUs used on each node v, denoted by $\theta_v(\gamma)$.

We ensure that, for each service $k \in K$, we do not embed more services than requested with the following constraints:

$$\sum_{\gamma \in \Gamma_k} z_{k\gamma} \leq n_k, \tag{4.2a}$$

where n_k is the number of requests requiring the same service k.

For each link $e \in E$, we define its capacity constraints as:

$$\sum_{k \in K} \sum_{\gamma \in \Gamma_k} \delta_e(\gamma) z_{k\gamma} \leq B_e, \tag{4.2b}$$

And for each node $v \in V$, we define its capacity constraints as:

$$\sum_{k \in K} \sum_{\gamma \in \Gamma_k} \theta_v(\gamma) z_{k\gamma} \leq C_v. \tag{4.2c}$$

Finally, the objective function is written as:

$$\max \sum_{k \in K} \sum_{\gamma \in \Gamma_k} z_{k\gamma}. \tag{4.2d}$$

Pricing Problems

Each service k has its own embedding sub-problem. The problem is similar to the compact formulation, so we reuse the same notations for the variables.

— $x_e^l \in \{0, 1\}$ indicates if the virtual link $l \in L_k$ of the service is routed through the physical link $e \in E$.

— $y_v^f \in \{0, 1\}$ indicates if the VNF $f \in F_k$ of the service is instantiated in node $v \in V$.

The first set of constraints ensures that each VNF $\forall f \in F$ is embedded on a physical node:

$$\sum_{v \in V} y_v^f = 1. \tag{4.3a}$$

Unlike the compact formulation, we write these constraints as equalities to make sure that the service graph is embedded.

The second set of constraints ensures that each node $v \in V$ can host up to one VNF of the service:

$$\sum_{f \in F_k} y_v^f \leq 1 \tag{4.3b}$$

We ensure flow conservation for each node $v \in V$ and each VL $l = (f_i, f_j) \in L_k$ with the constraint

$$\sum_{e \in \omega(v)} x_e^l - \sum_{e \in \omega^-(v)} x_e^l + y_v^{f_i} - y_v^{f_j} = 0. \tag{4.3c}$$

The overall delay of the service is ensured with the constraint:

$$\sum_{e \in E} D_e \sum_{l \in L_k} x_e^l \leq D^k, \tag{4.3d}$$

and the delay between each virtual link $l \in L_k$ with the constraint:

$$\sum_{e \in E} D_e x_e^l \leq D_k \tag{4.3e}$$

Pricing Objective Function

The pricing problems feed improving columns to the master problem. Improving and non-improving columns differ in their reduced costs as the reduced cost of a variable indicates the improvement of the objective function if the variable enters the solution. The goal of the pricing problem is to find the columns with the best reduced cost. If all variables have a null reduced cost, then the CG algorithm has converged.

We can obtain a variable's reduced cost formula from the dual of the master problem. If we define by π the dual values corresponding to the constraints of the primal problem (and let the exponent define the corresponding constraint), the dual problem is formulated as follows:

$$\min \quad \sum_{k \in K} n_k \pi_k^{(4.2a)} + \sum_{e \in E} B_e \pi_e^{(4.2b)} + \sum_{v \in V} C_v \pi_v^{(4.2c)} \tag{4.4a}$$

$$s.t. \quad \pi^{(4.2a)} + \sum_{e \in E} \delta_e(\gamma) \pi_e^{(4.2b)} + \sum_{v \in V} \theta_v(\gamma) \pi_v^{(4.2c)} \geq 1$$

$$\forall k \in K, \forall \gamma \in \Gamma_k \tag{4.4b}$$

Columns in the master problem become constraints in the dual problem and are also in exponential numbers. Similarly to the CG algorithm, the row generation algorithm starts from a reduced problem and searches for any violated constraints. Given a dual solution $\bar{\pi}$, the separation problem of the dual searches an embedding that violates constraints (4.4b), *i.e.,* any embedding γ such that

$$\bar{\pi}^{(4.2a)} + \sum_{e \in E} \delta_e(\gamma) \bar{\pi}_e^{(4.2b)} + \sum_{v \in V} \theta_v(\gamma) \bar{\pi}_v^{(4.2c)} < 1. \tag{4.5}$$

Going back to the embedding sub-problem for a given service k, defined by con-

straints (4.3), the objective function becomes

$$\min \sum_{e \in E} \bar{\pi}_e^{(4.2b)} \sum_{l \in L_k} B_l x_{el} + \sum_{v \in V} \sum_{f \in F_k} C_f \bar{\pi}_v^{(4.2c)} y_{vf}. \tag{4.6}$$

If the optimal value of this problem is strictly less than $1 - \pi_k^{(4.2a)}$, we know that adding the corresponding embedding into the master problem will improve the solution. Otherwise, no embedding can improve the master problem.

4.4 Proposed Solution

So far, we tried to address the placement problem using ILP and CG. The evaluations in the experiment section show that, although CG performs pretty well on small graphs of the services and the networks, it is not scalable for the bigger graphs. This is where our solution plays its role. For simplicity, we use *network* and *service* to refer to the topology graphs of the Substrate Network and the Service Request.

4.4.1 BnB-Based Service Placement

BnB is one of the most popular paradigms of algorithm design used for solving combinatorial optimization problems with exponential complexity.

BnB allows a complete search over the feasible solution space and avoids progressing in unfeasible branches by pruning the tree. We propose a BnB approach over a search tree (Fig. 4.1). We designate a node of the search tree as a *state* to differentiate it from a network node. Each state carries placement information, as well as a complete image of the network (remaining resources of the nodes and links). A terminal state contains the complete placement of the service.

Generating Search States

Starting at the root state of the tree, we select the first VNF of the service and we generate a corresponding sub-states (having a parent-child relationship) for each network node that can provide the required resources. Each state contains a complete image of the network, *i.e.,* the remaining resources after placing the VNF over the corresponding node.

Once a VNF is placed, we can place its outgoing and incoming VLs if both of their end-points are placed whether by the sub-state or its ancestors. In that case, we search

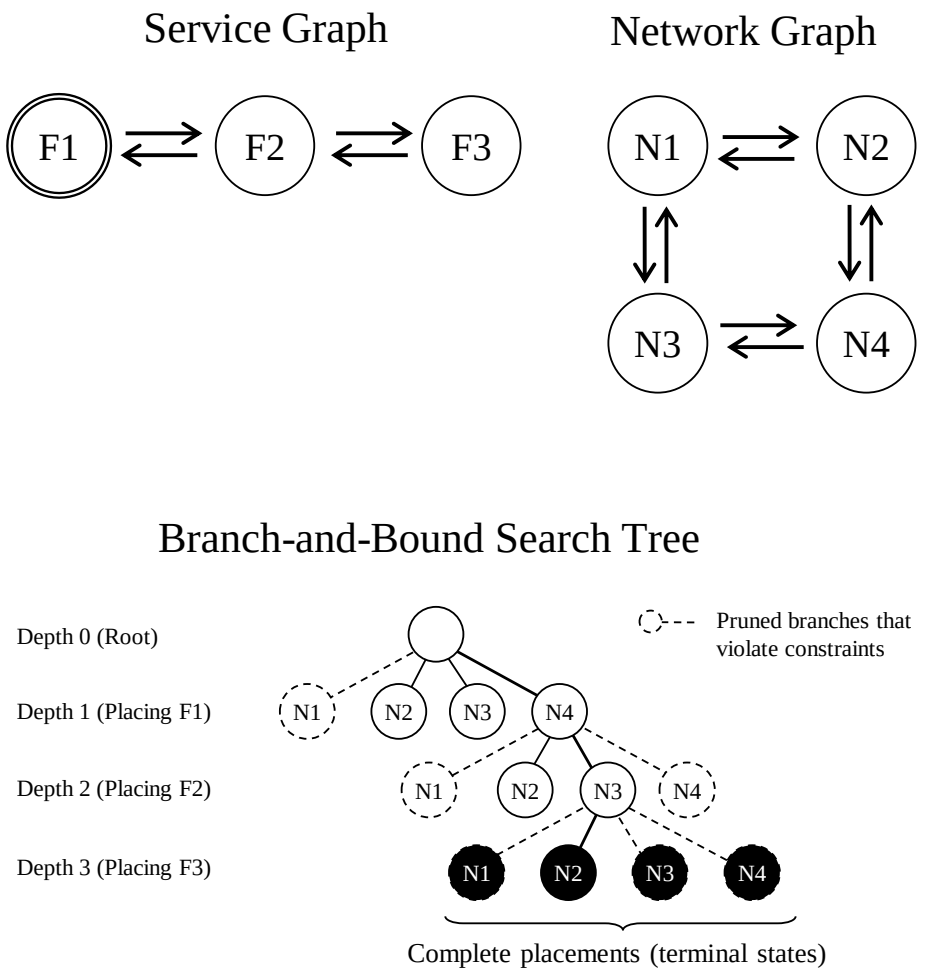


Figure 4.1 – A sample BnB search tree, placing a service with 3 VNFs over a network with 4 nodes

for one of the shortest paths between both end-points to embed the VL (considering the required bandwidth and latency), and update the network image with the resource used for the VL.

VNF Selection Order

The search tree evolves by continuing to select a VNF from the service and then placing it over the network nodes, generating their corresponding sub-states. To decide which VNF to be selected, we use the depth of the corresponding state and an ordered list of the VNFs of the service. In order to generate the list of VNFs, we traverse the service in Breadth-First Search (BFS), starting from the entry point ("first" VNF) of the service.

Traversing in BFS allows us to have a partial placement of a connected sub-graph of the service on each state, making it possible to be evaluated against constraint violations. The root state of the search tree is located at depth zero, so the states of the first depth contain partial placements of the first VNF of the list. Consequently, the last VNF of the list is placed by the states of the last depth. Thus, if we are placing a service with N number of VNFs over a network with M number of nodes, the depth of the search tree would be N, and the maximum branching factor would be M.

Note that our solution does not impose any constraints for sharing a previously placed network function between services. In case that we want to share and use exclusively an already placed network function of the same type of the currently placing VNF (whether it is a PNF or a VNF). Since the already placed network function has a known network location (network node), the other branches corresponding to other network nodes are pruned (there are several works considering the VNF sharing, like [35]).

As it is shown in Fig. 4.1, the branches are pruned because of a violation of constraints like:

— Not being able to provide enough node resources for the current VNF

— Not being able to find a path for the VLs connecting the currently placed and already placed VNFs because of the lack of the bandwidth

— Not being able to find a path that could satisfy the overall E2E latency requirement of the service

— Having placed the current VNF over a network node that we already placed another VNF of the same service over that node (VNE constraint)

Search Strategy

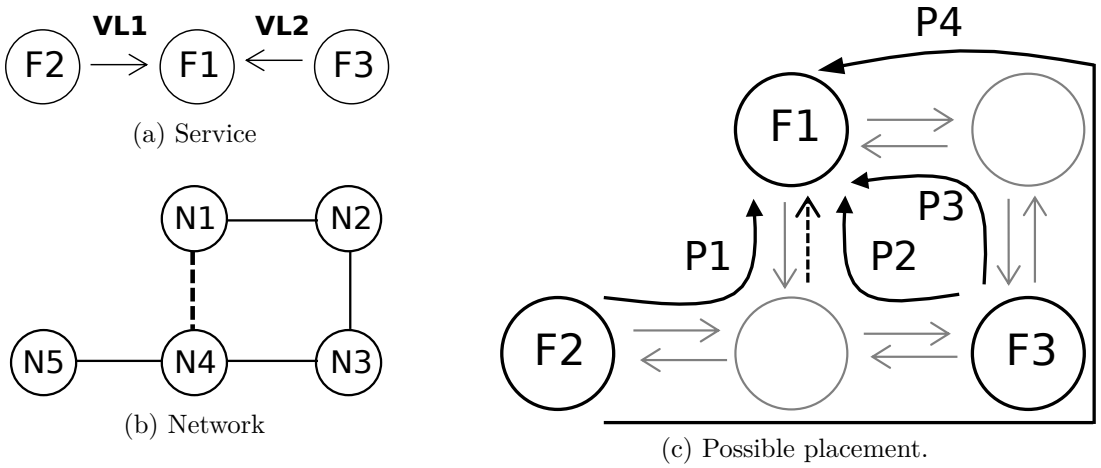
After expanding the root state (generating all of its sub-states), we continue the search by selecting and expanding the states until we succeed to find a terminal state, or we fail by reaching a timeout. When we expand a state, we evaluate the sub-states against the constraints, and then we store them in an ordered list called the fringe (we do not add the violating states to the fringe, this is how we prune the violating states from the tree). The next state to be expanded is chosen from the head of the fringe. The sequence of the states kept in the fringe specifies the direction of the search and it is determined by a search strategy. A search strategy specifies a traversal over the search tree (the order of visiting the states) through the fringe. The fringe can either be a queue (*e.g.,* in BFS), a

stack (*e.g.,* in Depth-First Search (DFS)), or an ordered list (*e.g.,* in A*). Our ultimate goal is to investigate the influence of applying various search strategies on the objective of SA (*i.e.,* the maximum number of service requests that can be accepted). Accordingly, we will propose several search strategies.

4.4.2 Node-Based vs. Link-Based BnB

Until now, we tried to explain the structure of our BnB by focusing on a node-based placement approach. The states represented the possible placements for the VNFs over the nodes, while the VLs were placed in a known order. Since all of the network nodes are considered candidates for placing each VNF, the order of placing VNFs does not violate the completeness of our BnB (completeness guarantees to return a solution if at least one solution exists). But fixing the order of placing VLs and not considering the other possible sequences might violate the completeness. That is to say, it is possible to fail to find any placement (when all of the branches are pruned and we can not reach a terminal state), while we could have found a placement only by considering another order of placing VLs. Although this occurs only when the resources are scarce, it might happen.

The mentioned problem is illustrated in Fig. 4.2, in which we try to place a service (Fig. 4.2a) over a network (Fig. 4.2b). As shown in Fig. 4.2c, we have placed F1, F2, and F3 over N1, N5, and N3, respectively. Suppose that we decided to place VL1 before VL2. We have two options to place VL1 (over the path of P1 or P4). According to the algorithm, we will choose P1 since it is the shortest path. Suppose that, by placing VL1 over P1, we use up the entire available bandwidth of the network link connecting N4 to N1. To place VL2, since the network link connecting N4 to N1 is taken by VL1, we cannot place VL2 on P2, but it is possible on P3 since it is available. Now suppose we decide to place VL2 first, and we decide to place it on P2 (both P2 and P3 are shortest-path candidates). Accordingly, if VL2 uses up the entire available bandwidth of the network link connecting N4 to N1, we will have to place VL1 over P4, which uses significantly more bandwidth and may violate latency requirements, leading to placement failure. By adopting a similar approach and redefining the BnB structure on links instead of nodes, we can consider all possible sequences of VLs placement.



(a) Service

(b) Network

(c) Possible placement.

Figure 4.2 – The need for the link-based placement approach

4.4.3 DFS-Based Search Strategy

Search strategies, regarding their types of traversal, are generally categorized into two classes: BFS-based and DFS-based. Note that BFS-based strategies include all of the traversals along with the breadth or a cost function, regardless of being informed or uninformed. DFS-based strategies traverse along with the depth of the search tree in order to quickly reach the terminal states. DFS Bandwidth Optimized (DBO) is another strategy that allows us to obtain high-quality near-optimal placements as fast as possible. In DBO, the fringe is a stack. The search tree evolves by popping a state from the fringe, expanding the state, sorting the sub-states by the amount of used bandwidth, and then pushing back the sub-states into the fringe, until reaching a terminal state.

4.4.4 Best-Fit-Based Search Strategies

BF is one of the most well-known heuristic algorithms for service placement. BF sorts the nodes and places the VNF over the node containing the maximum amount of available resources. The placement fails to place a service as soon as it could not find any node to provide the required resources of a VNF of the service.

Looking for a way to see to what extent we can improve and benefit from the BF, we imagined the possibility to revise the previously placed VNFs. Consequently, we realized that our BnB structure is a perfect fit to put our idea into practice. By tweaking the DBO to sort the generated sub-states according to their amount of available node resources,

instead of the used bandwidth, we can obtain another proposed search strategy called Enhanced Best-Fit (EBF).

4.4.5 A* Based Search Strategies

A^* is amongst the most popular AI informed search algorithms due to its completeness, optimality, and optimal efficiency. It is considered the best solution for many problems in computer science. A* traverses the search tree according to f-costs, which for an arbitrary state of n is calculated by the sum of a cost function and a heuristic:

$$f(n) = g(n) + h(n) \tag{4.7}$$

The states are stored in a list called fringe, an ordered list based on the f-costs. f is an estimation of the cost of an optimal solution from the root to a target state, the heuristic function h estimates the cost of an optimal solution from the current state to a target state, and g is the real cost of the path from the root to the current state. The optimality and completeness of A* are proved if we choose an admissible and consistent heuristic function [100].

ABO

In A* Bandwidth Optimized (ABO), we use A^* algorithm by focusing on bandwidth usage optimization. A heuristic that estimates the cost of the placement can be defined as the cost of placing each unplaced VL over a path containing exactly a single network link (since we need at least one network link to accommodate each VL). So, we consider the total amount of required bandwidth by all of the unplaced VLs as h, and g represents the total amount of bandwidth used for all of the placed VLs.

Theorem 1. *Our heuristic function is admissible.*

Proof. A heuristic function is admissible if it never overestimates the cost to the target state [100]. If we can place the source and the destination of a VL over two adjacent network nodes, the cost of placement will be equal to what our heuristic estimates. Otherwise, we need a path of several links to accommodate a VL, which means the cost would be more than what our heuristic estimates. Thus, our heuristic never overestimates the cost to the target state, so our heuristic function is admissible. □

Theorem 2. *Our heuristic function is consistent.*

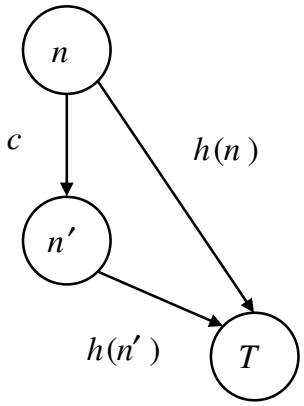


Figure 4.3 – Illustration of the triangle inequality

Proof. A heuristic function is consistent if the f-cost never decreases along a path from the root state to any arbitrary state [100]. If n' is a successor state of n and c is the cost of placing unplaced VLs on n that are already placed on n' ($c = g(n') - g(n)$), we need to prove the triangle inequality of $h(n) \leq c + h(n')$, as illustrated in Fig. 4.3. We label the estimated cost of transition from n to n' as c' ($h(n') = h(n) - c'$). Knowing that the estimated cost of placement on n' is less than the estimated cost on n (since we have fewer unplaced VLs on n'), we just need to prove that $c' \leq c$. Since our heuristic does not overestimate the cost, this inequality is always true, *i.e.,* our heuristic is also consistent. □

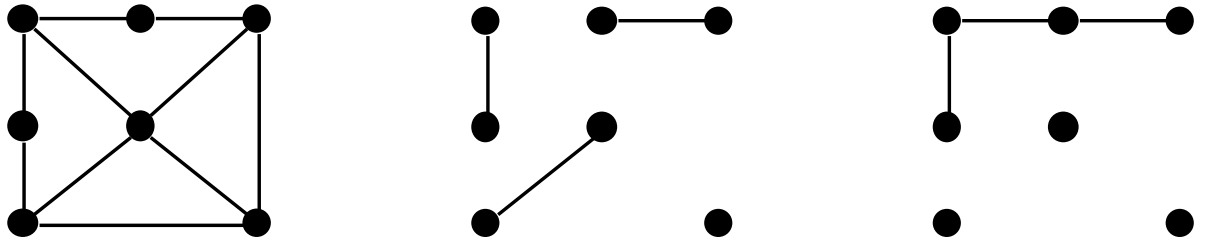
By proving Theorem 1 and Theorem 2, we can tell that our A^* search is complete and optimal. Note that, the optimality guarantees that if A^* succeeds to find a placement, no other placement exists using less bandwidth than the found one.

FABO

Along with the optimality, increasing the chances of accepting new service requests is also important to consider.

Network fragmentation is a critical issue which leads to a remarkable service rejection, and it is hardly addressed in the literature. It happens when the placement algorithm tends to place new service requests over the same nodes and links as long as they can provide the required resources, resulting in too many disconnected sub-networks. It is even more likely to happen if the algorithm is optimal and deterministic.

To illustrate this problem, we give an example. Fig. 4.4(a) shows a network topology, where each link provides the same units of bandwidth. Fig. 4.4(b) and 4.4(c) represent



(a) (b) (c)

Figure 4.4 – Network fragmentation problem

the remaining links as a result of placing the same number of services by two different placement algorithms. Considering that we respect the anti-affinity constraint, although in Fig. 4.4(c), it is possible to place the services containing 2, 3, or 4 numbers of VNFs, in Fig. 4.4(b), we can only place the services having two VNFs, reducing the chances of service acceptance drastically.

Fortunately, network fragmentation can be mitigated by taking into account the relevant considerations in the placement algorithm.

We address this problem by modifying our approach and proposing Fair A* Bandwidth Optimized (FABO) to establish a uniform distribution of the bandwidth, and we call it a fair placement. On the one hand, it tries to avoid network fragmentation by sorting the states of the fringe according to the number of disconnected sub-networks which have at least two network nodes. On the other hand, it tries to make a fair distribution of the load over the network links by sorting the states of the fringe based on the variance of the remaining bandwidth of the network links.

In a nutshell, the fringe is sorted with multiple criteria. The first criterion is the f-costs since FABO is defined on top of the optimal ABO. The second is the number of disconnected sub-networks, and the last one is the variance of the remaining bandwidth.

4.4.6 Parallel Integrated Search Strategy

To maximize service acceptance, it is inevitable to minimize service rejection. Finding optimal placements and preserving the resources as much as possible is necessary, but being unable to find an optimal placement within the time constraint, leading to service rejection, is not coherent with our objective.

Accordingly, it is essential to have backup strategies to find near-optimal placements as fast as possible. To escape from service rejection, we need to try multiple strategies.

Parallel execution of the strategies allows each strategy an equal opportunity over time. Parallel Integrated (PI) is our another proposed search strategy that does not define a specific traversal, but it combines several strategies in order to obtain the best results in terms of service acceptance. In PI, we execute FABO, ABO, and DBO in parallel and wait until they finish (whether they succeed or reach the timeout). If multiple strategies succeed to find the placements, the one with the best quality is selected (*i.e.,* the placement which is found by FABO is preferred over the one found by ABO, and similarly, ABO's result wins over DBO's).

4.5 Experimentations

4.5.1 Parameters and Steps of Evaluation

The evaluations are executed on a system with an Intel Core i7-3687 CPU and 8GB of RAM. Our implementations are made in Java running on Windows 10. We set a timeout of 2 seconds for placing a service in all of our evaluations (*i.e.,* if a strategy can not find a placement within the timeout, it fails).

We perform our evaluations over the BT-Asia-Pacific, BT-Europe, and BT-North-America network topologies, selected from the Zoo Topology dataset [101], with respectively 20, 24 and 36 nodes, and 62, 74 and 152 bi-directional links. We consider daisy chain, ring, and star topologies for service graphs having 3 to 10 VNFs and bi-directional VLs. Each VNF requires a single unit of CPU and a single unit of storage, and each VL requires from 1 to 10 bandwidth units. We use the SAMCRA [102] routing algorithm which considers QoS metrics to place a VL over a network path.

Our evaluation begins by initializing the network nodes/links with their maximum available resources. Then, iteratively, we generate service requests one by one based on the specified topology, size, and other requirements. Then, we try to place them using the specified strategy within the timeout. On each iteration, if we find a placement for the requested service, we apply the placement over the network by reserving its required resources, and then we start a new iteration. Otherwise, we terminate the evaluation and report the number of services successfully placed.

4.5.2 Assumptions

To compare our strategies, we will introduce some assumptions that are not necessarily required in real scenarios but they help to highlight the behavior of the strategies.

No Constraints for Overall Delay

The overall delay is systematically verified in the BnB, however, to enhance clarity in distinguishing between strategies, our focus lies on the allocation of resources rather than solely on overall delay.

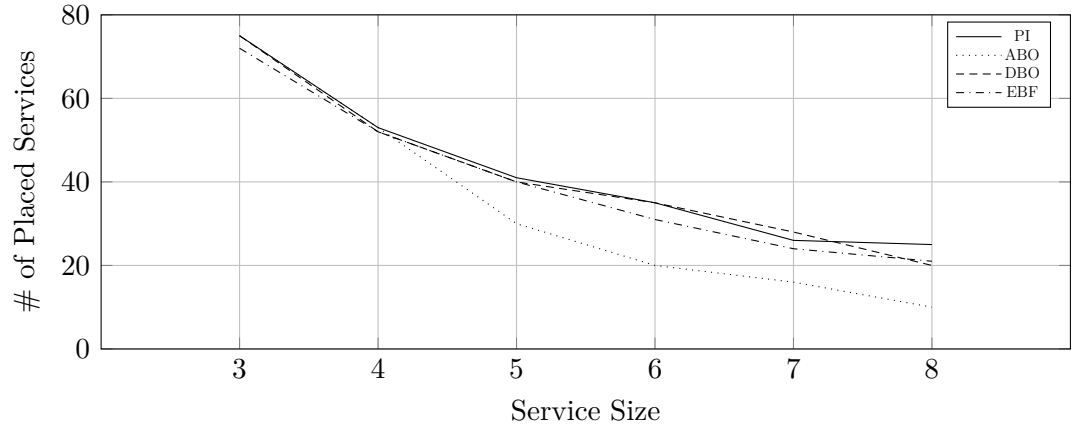
No Constraints for Node Resources

As our default configuration, we consider placing services with daisy chain topology, with VNFs requiring one unit of CPU and storage, and one unit of bandwidth for VLs, over BT-Europe network. To illustrate the impact of node resource limitation on the placement strategies, we considered 10, 20, and 40 available units of CPU and storage resources for each network node. Fig. 4.5 shows that all placement strategies act in a similar way when nodes have a small number of resources. The effectiveness of the strategies becomes clearer when network nodes have more resources. Therefore, from now on, unless mentioned otherwise, we consider an unlimited amount of CPU and storage for each network node and 10 units of bandwidth for each network link, at the beginning of the evaluation.

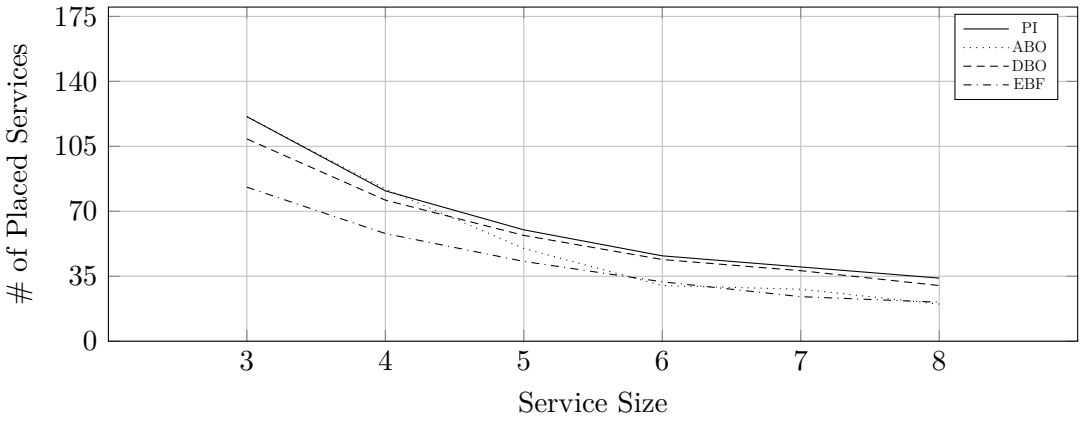
Anti-Affinity Rule

Affinity and anti-affinity rules in the cloud-computing provide a mechanism for establishing a trade-off between the performance and the reliability of the realized service. Affinity asserts putting VNFs on the same network node for increasing inter-networking performance. While, anti-affinity is used to improve the reliability and the availability by preventing certain VNFs of the same service from sharing the same physical resources in order to reduce the impact of a single network node failure [103], [104]. Moreover, placing the VNFs of the same service over the same network node (*a.k.a.* VNF consolidation) may cause severe performance degradation (*a.k.a.* VNF interference), which is not acceptable for some QoS-sensitive 5G use cases [105].

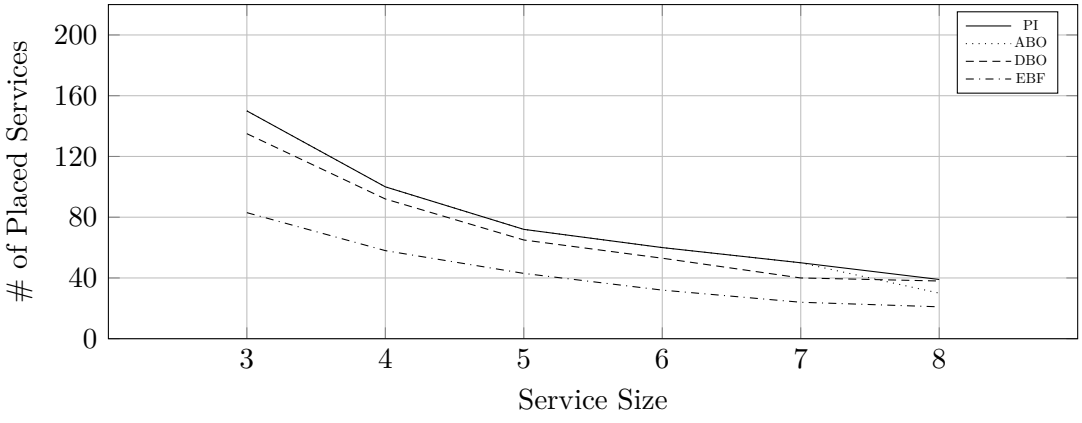
We respect the anti-affinity constraint, which is usually considered in the Virtual Network Embedding (VNE) problem in the literature. According to this constraint, placing any two VNFs of the same service over a same network node is not allowed (*i.e.,* all of



(a) nodes with 10 units of maximum available resources



(b) nodes with 20 units of maximum available resources



(c) nodes with 40 units of maximum available resources

Figure 4.5 – Illustration of node-resource limitation impact

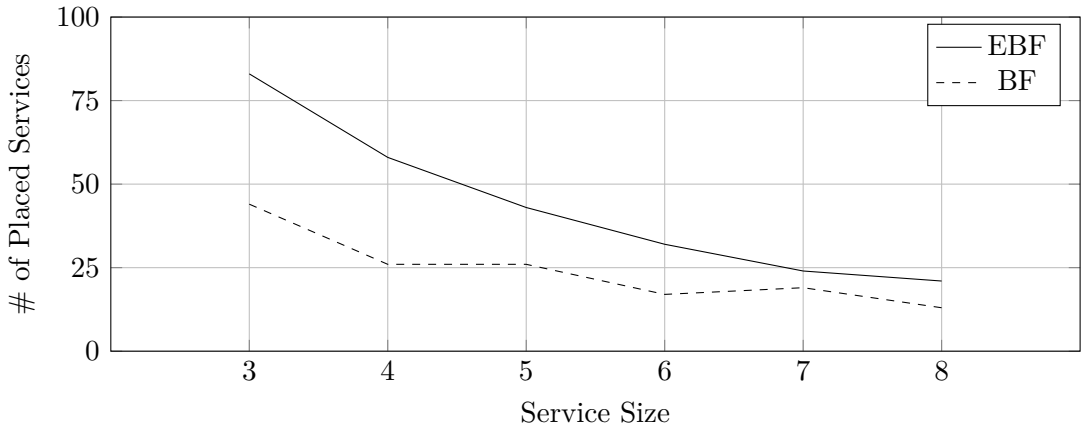


Figure 4.6 – BF vs. EBF

the VNFs need to be placed on separate network nodes). Not considering this constraint results in a remarkable relaxation of the placement problem, so that the gain of using different placement approaches becomes marginal.

The anti-affinity is often partially considered in a service. But since placing a single VNF over a single network node does not make a major difference in terms of the complexity in comparison to placing a group of VNFs (that need frequent communications) over a single node, we consider anti-affinity for all of the VNFs of the service (to confront to the maximum complexity which this constraint can create).

4.5.3 BF vs. EBF

Fig. 4.6 shows that we can place on average almost 2 times more services with EBF, compared to BF. We tried to place daisy chain services with different sizes over the BT-Europe network.

4.5.4 The Advantage of Integrating the Strategies

To evaluate the integration of strategies, we can imagine a strategy that executes DBO when ABO fails to find an optimal placement (let's call it ADBO).

Table 4.2 shows the number of placed services, and the percentage of the total remained network bandwidth at the end of the placement for ABO, DBO, and ADBO strategies.

In general, ABO places more services and can save more remaining bandwidth. But, for services with 8 VNFs, although ABO saves a significant amount of bandwidth (43.24% vs.

Table 4.2 – The Advantage of Integrating the Strategies

Size	# of Placed Services			% of Remained BW		
	ABO	DBO	ADBO	ABO	DBO	ADBO
3	180	173	180	2.7	2.1	2.7
4	115	101	115	5.4	9.4	5.4
5	78	78	78	13.5	8.3	13.5
6	60	57	60	18.9	12.9	18.9
7	55	46	55	6.7	12.1	6.7
8	**30**	**38**	**39**	**43.2**	**7.0**	**17.5**

7.03%), it places fewer services (30 vs. 38). Here, ABO fails to find an optimal placement within the time limit, and optimality is sacrificed over time, while, their combination (ADBO) places more services and leaves the remaining bandwidth of 17.5%. The time restrictions on online placement argue for the integration of strategies to minimize service rejection. Accordingly, we introduced PI by parallel integration of FABO, ABO, and DBO.

4.5.5 Service Acceptance

Extensive evaluations were performed. Considering 10 different values of VL bandwidth requirement, 3 types of service topologies, 8 service graph sizes, and 6 strategies, we performed 4320 evaluations (1440 evaluations for each of the three network topologies). The results confirm that PI is almost always superior to the other strategies, making us able to measure them all against PI (Table 4.3). We used quartiles (Q1, Q2 (Median), Q3) as well as min-avg-max, in order to represent the results. For example, an improvement of 494% on average means that PI places 5.94 times more services in comparison to BF considering the same evaluation parameters. Beyond these satisfactory results in terms of average, the max column shows that PI can place 25 to 49 times more services, when the placement problem becomes so complicated that DBO and EBF could not find any placement or found only 1 or 2.

4.5.6 The ILP and CG Results

We compare, in Table 4.4, the PI strategy with the offline solutions provided by the ILP and the CG algorithm. Both mathematical formulations were solved using the Gurobi solver. Solving the ILP becomes more difficult as the size of the services increases; it takes more than three hours to find the optimal solution for daisy chains of size 5. For larger

Table 4.3 – Service Acceptance Improvements (in percent), gained by using PI instead of other strategies

Strategy	Q1	Q2	Q3	Min	Average	Max
FABO	0	19	50	0	**115**	3800
ABO	0	15	33	-14	**26**	300
DBO	12	27	50	-14	**150**	4800
EBF	100	112	173	0	**208**	2400
BF	217	325	600	90	**494**	2400

Table 4.4 – Evaluation of the PI strategy with the ILP and the CG algorithm

Size	ILP		CG & BnB		PI Strategy	
	#	Time	#	Time	#	Time
3	185	757	185	6	185	30
4	123	1857	123	22	121	29
5	92	12567	91	71	90	26
6	-	-	73	246	72	28
7	-	-	60	571	59	31
8	-	-	49	1113	47	29

instances, we rely on the CG algorithm to provide the solution and its upper bound. We see that our PI strategy finds close-to-optimal solutions, with an average gap of 2%.

Note that we can't expect an online placement solution like PI to reach the global optimum of the offline problem. It should also be noted that PI achieves this performance while keeping the computation time around 30s. Even though the CG algorithm shows better scalability in comparison to ILP, it still needs about 20 minutes to find a solution for daisy chains of size 8.

Furthermore, we were interested to see to what extent our placements found by the PI strategy are optimal. Note that the placements found by FABO and ABO are guaranteed to be optimal, but the ones found by DBO are not necessarily optimal. By performing a complete evaluation over the BT-Europe network topology, we witnessed that 89% of the placements were found by FABO and ABO, indicating that at least 89% of the placements are optimal.

Fig. 4.7 depicts a general comparison between different strategies except for PI. The behavior of the PI strategy depends on the timeout. In fact, by considering a very short period as a timeout, the PI strategy acts like DBO; by considering longer periods, it acts

Table 4.5 – Evaluation of the other strategies

Size	ABO		DBO		EBF		BF	
	#	Time	#	Time	#	Time	#	Time
3	185	1.8	173	0.9	82	0.4	44	0.2
4	106	1.8	102	0.7	58	0.4	26	0.2
5	82	1.4	78	0.6	43	0.4	26	0.3
6	61	3.8	58	0.5	32	0.5	17	0.2
7	52	3.5	46	0.5	25	0.4	19	0.2
8	31	3.6	37	0.5	20	0.4	13	0.2

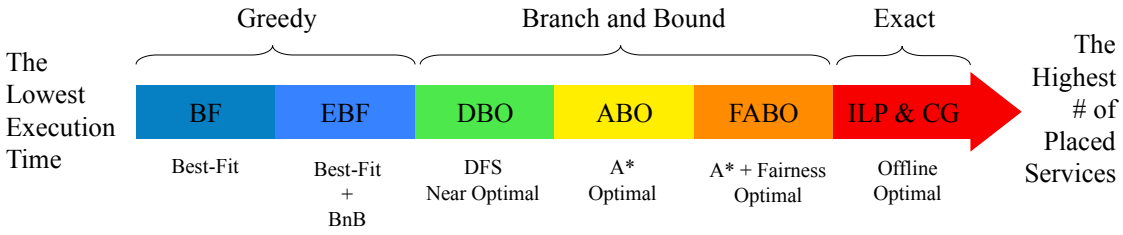


Figure 4.7 – A general comparison between different strategies

like ABO; and if we give it enough time, it acts like the FABO strategy.

The execution time of the other strategies are shown in Table 4.5 for more information.

4.5.7 Fair-Placement and Network Defragmentation

Up to this point, we have considered a fixed size for the service requests during an evaluation. Yet, in reality, the arriving service requests can contain different number of VNFs and VLs with different resource requirements. We try to generate random services, and to guarantee the reliability and the repeatability of the results, we perform this scenario many times with different seeds of randomness (i.e., performing Monte Carlo experiments [106]). Since the number of placed services (our random variable) is bounded (because our resources are limited), we can rely on the convergence in terms of the average of the outcomes of our random experiments. In each random experiment, we start with generating a daisy chain service with a random number of VNFs (with a uniform distribution) in a range of $[3, 8]$, and continue placing them, until the specified strategy fails.

We want to compare the PI strategy, which performs a fair placement and avoids

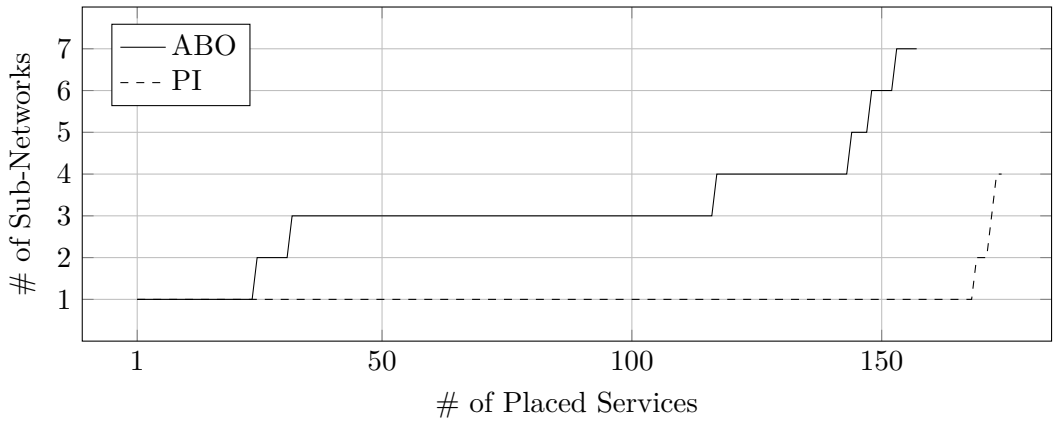


Figure 4.8 – Network Fragmentation on BT-North-America

network fragmentation, with the ABO strategy, which only performs optimal placement. Fig. 4.8 represents a single random experiment over the BT-North-America network. The number of created sub-networks confirms the effectiveness of PI, not only in keeping the network's integrity until almost the end of the simulation (keeping a single piece of sub-network) but also in its capability to place more services than ABO.

If we take the sequence of the number of sub-networks on the experiment related to the Fig. 4.8, as we continue placing the services over the network, the average of this sequence will be 3.09 for ABO, and 1.06 for PI strategy.

We performed 100 random experiments, with different seeds of randomness for each of the strategies and for all of our three networks. Fig. 4.9 shows the min-avg-max over the average number of sub-networks during each random experiment. The average obtained using ABO on BT-Asia, BT-Europe and BTN-America is 1.18, 1.38 and 1.82 (note that it increases according to the size of the network), while it is near 1.02 using PI for all of the results obtained for all these networks.

The number of placed services in 100 random experiments over the BT-Europe network topology is represented in Fig. 4.10. Fig. 4.11 demonstrates the cumulative average of the results of Fig. 4.10. The superiority of the PI strategy can be witnessed not only by the convergence of the cumulative average of PI towards 80.5 (95% confidential interval with 0.9% margin of error) and ABO to 68.4 (95% confidential interval with 1.7% margin of error) but also the fact that PI could almost always place more services than ABO considering each random experimentation. So, we can claim that PI strategy can place almost 18% more services over the network on average in comparison to ABO strategy by performing a fair placement.

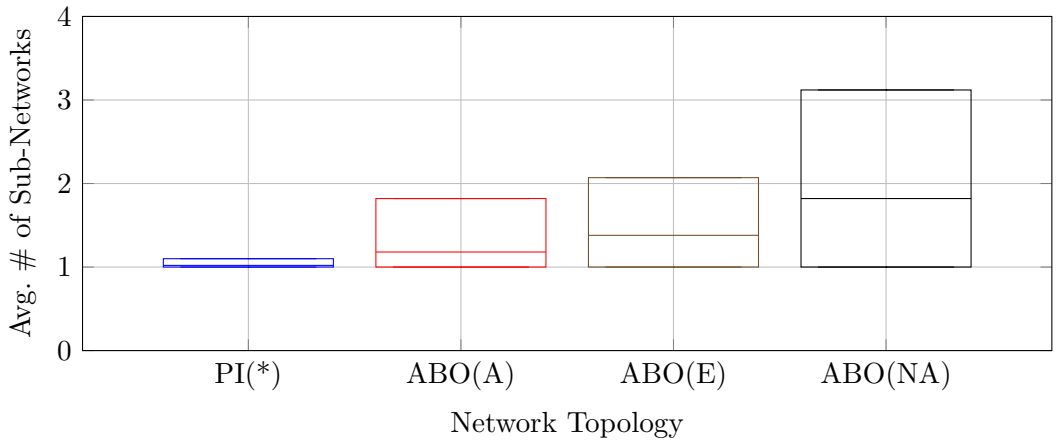


Figure 4.9 – Network Fragmentation on different networks including BT-Asia, BT-Europe, and BT-North-America

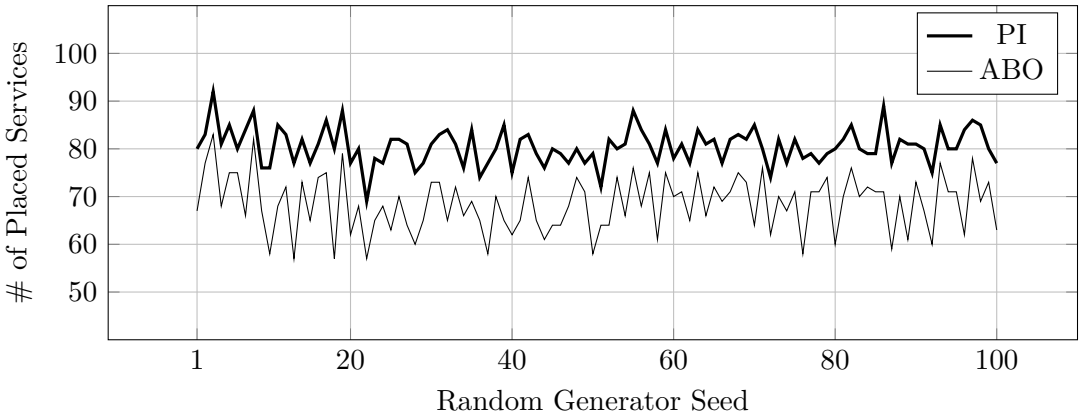


Figure 4.10 – The number of placed services in each random experiment

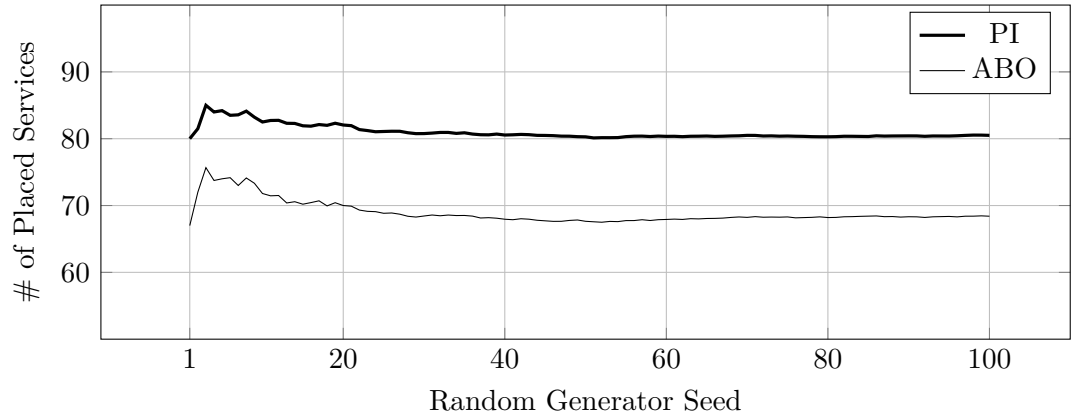


Figure 4.11 – The cumulative average of the number of placed services in each random experiment

4.5.8 Comparison with ML-Based Approaches

The authors in [31] propose a DRL-based approach for maximizing the SA, considering the QoS requirements. They compare their results with similar configurations as our approach (like using Bt-Europe topology and similar resources), with the FF algorithm (FF is a simplified version of the BF algorithm, in a way that it does not sort the network nodes before placing each VNF). They defined the possible states, the actions and the rewards for the agent of the DRL to find the placements for the VNFs of a service. They used Deep Deterministic Policy Gradient (DDPG) in order to enhance the performance of DRL agent. They demonstrated almost 30% of improvements in the number of accepted services in comparison to the FF algorithm. [32] is another work proposing DRL and with RGCN for maximizing the service acceptance, and they showed almost 20% of improvements in the number of accepted services in comparison to the BF algorithm.

In ML-based approaches, the placements is made in two phases. At first, the model is trained by the making random placements to let the agent learn to maximize the number of placements. Then, the model is evaluated to see how many placements it can make. Although the obtained results of the these approaches are not as good as the results of our proposed approach, there are particular advantages by using DRL. In fact, the majority of DRL-based approaches can generate results super fast after the training phase, making it ideal for large scale placement problems.

4.5.9 Reproducibility

Reproducibility is one of the most important criteria, especially in evaluating a time-based algorithm when the algorithm uses several parallel threads (like our strategies). In other words, since we put the time constraint on our placement algorithm, if we run our experiments with another machine with different computing resources, the results could be different. Even on the same machine, the results could be slightly different each time we execute the same evaluation. The mentioned problem comes from the fact that the operating system might allocate different CPU time for a process or a thread over a fixed period based on its CPU power and current CPU load.

To overcome this problem, we only need to translate our time constraint into the constraint of the max iteration count. As mentioned before, our placement algorithm is performed iteratively, picking a state from the fringe and expanding it on each iteration. To guarantee the repeatability of our evaluations, we need to know how many iterations are

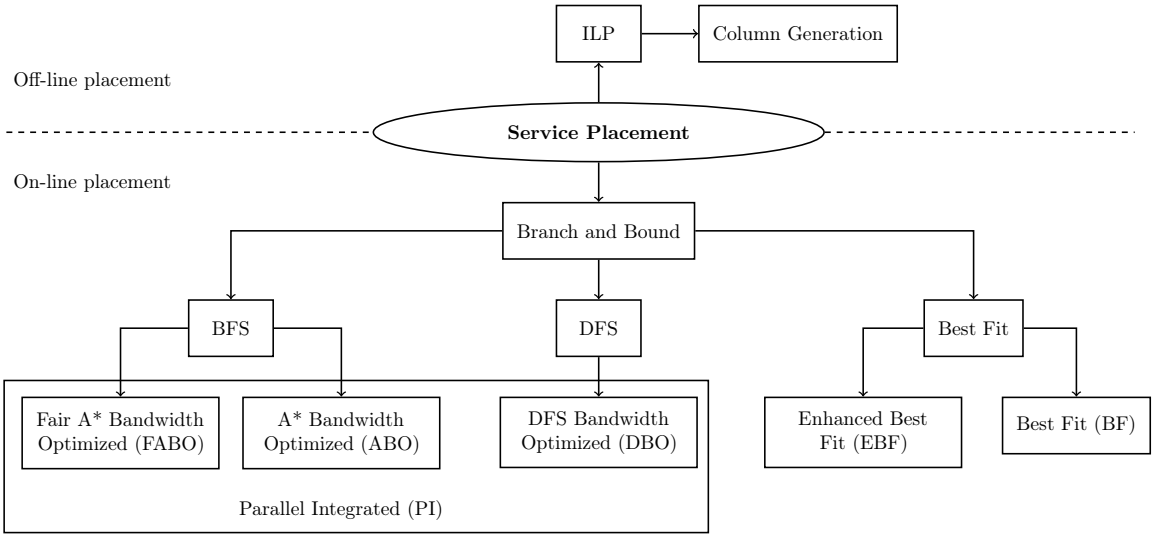


Figure 4.12 – Overview of the proposition

performed approximately during our time constraint and set the corresponding maximum iteration count instead of time. Note that this translation is essential for a valid comparison of different strategies in our evaluations.

4.6 Summary

Although the VNF placement problem has been studied for many years, the need for an approach that could find a fair compromise between optimality and scalability still exists. Recent advances in AI and ML techniques have revolutionized many research domains. We proposed a solution based on a BnB structure, adapting AI search strategies, particularly A^*, capable of finding optimal placements considerably fast. Substantial empirical analysis has been made, and the results confirm a considerable improvement (494% on average) in comparison to the popular placement algorithm of Best-Fit. Fig. 4.12 summarizes our proposed strategies.

As a downside of our A^*-based strategies, we could mention the exponential memory complexity of $O(b^d)$ (b stands for branching factor and d for depth). Yet, since the online placement imposes a time constraint, we only need to ensure that the search procedure will finish before using up the entire memory, depending on the scale of the network and the service. In addition, since there are effective algorithms like Simplified Memory Bounded A^* (SMA*) to address it, we do not consider it a significant issue.

Chapter 5

Network Service Placement with Limited Node Resources

5.1 Introduction

The online placement of the network services, demanding strict E2E latency requirements, over the edge networks, with restricted computing resources considering the end user location represents a challenging problem which is worth investigating. This is the subject, which we try to address in this chapter.

To meet the required latency of the network services, we can not place them on the clouds far from the end users, where the resources are abundant. Some services must necessarily be kept at the edge of the network, where resources are more scarce.

Following our objective of SA, in an offline placement, although the complexity is not negligible, by a well-defined ILP model, we can perform an optimization to find the maximum number of accepted services (*a.k.a.* acceptance ratio). In online placement, although we do not know the service requests beforehand (consequently, we cannot optimize directly the number of accepted services), we can ensure that the placement of the current service request enhances the chances of accepting future service requests. This can be done by optimizing a cost function which is consistent with our objective.

Like the previous chapter, we respect the anti-affinity constraint, in which we do not accept placing any two VNFs of the same service over the same network node.

Even though the problem of the placement has been investigated in several related areas, to the best of our knowledge, the placement of the service requests with strict E2E latency on the edge network having limited node resources, considering user-location and anti-affinity is not yet addressed.

Generally, the proposed placement approaches try to optimize the cost of the resources or the QoS requirements like bandwidth and latency. A network service requires node resources (*a.k.a.* computing resources) for deploying its VNFs and link resources (*a.k.a.*

network resources) for realising its VLs. Supposing that we do not share the already placed VNF (like in [35]), and we need to provide what is required as node resources for deploying the VNFs, the optimization involves placing the VNFs as close as possible to the user-location and the other VNFs, to minimize the overall bandwidth usage and/or the E2E latency. We will show that this optimisation approach could have a devastating effect on edge networks with limited node resources and needs to be avoided. We will see that it can lead to draining the resources in the proximity of the user location, which results in service rejection. Thus, we do not follow an optimization approach toward the bandwidth/latency, however, we consider them as constraints that need to be respected.

The contributions of the current work can be summarized as follows. We model the problem as an ILP and resolve the model to obtain the global optimum (which can only be achieved in an offline scenario). We propose an online approach based on BnB, capable of achieving optimal and near-optimal results according to several cost functions and search strategies. We also explore several cost functions and demonstrate the inconsistency of bandwidth and/or latency optimization, and the benefits of a fair placement, with our objective of maximizing the SA.

The *fairness* of our approach can be explained in two ways. On the one hand, our achieved improvements are due to the fair distribution of the resources in our placements. On the other hand, our approach makes a fair compromise between the service provider and the network/cloud provider. The service provider prefers that the realized latency of their services be the lowest possible (ideally zero), whilst the network provider would rather it to be the highest possible (maximum acceptable), allowing them to push the placement of the services from the edge to the clouds where the resources are abundant and cheaper. The maximum acceptable latency for the realized services is where we can compromise to meet the expectations of both sides.

This chapter is organized as follows. First, we explore the related works in Section 5.2. We present our ILP and CG formulation of the problem in Section 5.3. Then, we provide a comprehensive description of our BnB approach in Section 5.4. In Section 5.5, we explore the evaluations to investigate the effectiveness of our proposed solution. We will conclude in Section 5.6.

5.2 Related Work

Studying the placement problem in existing research begins with the advent of NFV, and it is frequently encountered in different use cases, considering different categories of objectives.

Addressing the problem by exact resolution includes formulating mathematically an ILP or an MILP, and solving via the related tools.

In [28], the placement is investigated for the network services under multiple constraints to maximise the SA, initially by suggesting an ILP, and subsequently a heuristic. They consider bandwidth and latency for the links, and computing resources for the nodes on a star-centric edge/cloud network topology. Although strict requirements of E2E latency of the service requests and the anti-affinity rules are not studied, limited node resources were considered on the edge, and they demonstrated that a balanced allocation of the resources can improve the number of accepted services.

Jin *et al.* try to address VNF placement problem in [107], considering the resource shortage on the edge with latency guarantees. First, they formulate the problem in MILP to minimize the consumption of the computing resources (by sharing and reusing the already placed VNFs), and the bandwidth. Note that using an already placed VNF reduces the cost of placing a new VNF, but it may result in using more bandwidth and latency to access that VNF (if it is placed very far from the other VNFs of the service). Although they did not consider anti-affinity rules, they proposed an interesting DFS based algorithm for placing the VNFs and the VLs to obtain near-optimal solutions for an online placement scenario.

The placement problem on the edge is also investigated in [108], using MILP over batch-based service requests, minimizing the overall network latency. They show that their model can improve the acceptance rate of the services with strict latency requirements, via sharing the already placed VNFs and considering the affinity rules. They define an affinity matrix to identify the VNFs with high-affinity, that can be placed over the same network node. High-affinity is defined when two VNFs exchange a big amount of data flow, while low-affinity or anti-affinity is considered to allow critical VNFs to be placed on separate network nodes (in case of failure). They proposed a heuristic placement algorithm for big networks and a large number of requests, and they compared it with a RF algorithm.

Even though the ILP-based methods guarantee optimality, they might encounter issues concerning scalability. Performing an exact approach could require a long execution time

for achieving an optimal result, which could make restrictions for using in large-scale networks. This constraint poses a significant obstacle for online placements which may have constraints on the execution time.

Heuristic approaches are often retained when it comes to scalability. BF is one of the most popular placement methods, which places the VNF by sorting the nodes according to their resources and placing the VNFs in an iterative way. [29] studies heuristic methods (including the BF), and the authors propose an algorithm based on a multiple-level graph to find near-optimal results in a way that can scale well to the bigger networks. They evaluate the execution time, the acceptance ratio, and the average placement cost regarding the assigned resources.

In [79], an adaptive heuristic approach is proposed to maximize the total throughput of accepted requests in an edge/core cloud network, considering anti-affinity rules. They try to avoid VNF consolidation to avoid severe performance degradation (*a.k.a.* VNF interference), which makes it intolerable for some QoS-sensitive 5G use cases (*e.g.,* autonomous driving and 4K/8K HD video).

In spite of being able to achieve the results as quickly as possible, heuristics do not guarantee the quality. Meta-heuristic and evolutionary algorithms represent a different direction to solving the placement problem. In [109], a fast sub-optimal Tabu Search based approach is proposed to minimize the end-to-end latency and the overall deployment cost. Although they are effective in skipping local optima, these algorithms often are hindered by unpredictability, particularly regarding execution time, which is of utmost importance in online placement.

Sophisticated AI search algorithms and innovative breakthroughs in ML (especially in DRL) have recently caught the interest of the community. The authors in [110] propose a RL approach to address the online placement of the requests after formulating an ILP placement approach for low-latency IoT services on Multi-Tier Mobile Edge Networks to maximize the throughput and the SA ratio, but they do not take into account anti-affinity rules. Although novel DRL-based approaches are extremely fast, but on one hand, we might face the optimality issues. On the other hand, we might not be able to explain why these methods work well in certain conditions (*a.k.a.* explainability issues in DRL-based methods [111]).

Table 5.1 – Notations

Name	Description
$G = (V, E)$	Substrate network
V	Set of nodes of the network
E	Set of links of the network
$\omega^+(v)$	Outgoing neighboring nodes of $v \in V$
$\omega^-(v)$	Ingoing neighboring nodes of $v \in V$
B_e	Bandwidth available on the directed link from $v \in F_k$ to $j \in w^+(v)$
$H_k = (F_k, L_k)$	Virtual graph of service k
F_k	Set of VNFs to be placed for service k
L_k	Set of VLs between the VNFs of service k
$\omega^+(f)$	Outgoing neighboring VNFs of $f \in F_k$
$\omega^-(f)$	Ingoing neighboring VNFs of $f \in F_k$
B_l	Bandwidth requested between the two VNFs of $l \in L_k$
C_f	Computing resources requested by $f \in F_k$
C_v	Computing resources available at $v \in V$
D_e	Latency experienced on link $e \in E$
D_k	Latency requested for service $k \in K$
D_l	Latency requested between the two VNFs of $l \in L_k$

5.3 Model

The placement problem is defined as placing a service request graph on SN graph. A service graph $k \in K$, where K is the set of services, is indicated by a directed graph $H_k = (F_k, L_k)$, containing a set of VNFs F_k, and a set of VLs (connecting the VNFs) L_k, plus SLAs (meeting the QoS requirements). A VNF $f \in F_k$ requests a subset of resources (*e.g.,* CPU C_f), and each VL $l \in L_k$ demands an amount of bandwidth B_l. Likewise, a SN is represented by a directed graph $G(V, E)$ of its nodes V and links E. A node $v \in V$ has a resource capacity (C_v), and a link $e \in E$ has a bandwidth capacity of B_e, besides QoS parameters (*e.g.,* E2E latency). Finally, we consider that a VNF $f \in F$ can only be instantiated on a subset of nodes $V_f \subseteq V$ of the SN. In the case of a user location, this subset can be reduced only the nodes that users are located. The problem is to find the mappings of the VNFs and the VLs of the service requests into the network nodes and the network paths respectively, subject to the constraints. The notations are summarized in Table 5.1.

5.3.1 Embedding Decomposition

Similarly to our previous chapter, we exploit a decomposition method for solving the offline placement problem. We still consider the *embedding decomposition*, where each variable represents a possible embedding of the service onto the physical network. Here we try to fix the user location, by considering the user location as a VNF of the service which is fixed on a network node and does not require resource. Accordingly, we should change the pricing problem.

Once again, the number of embeddings is exponential and an embedding-based formulation demands an exponential number of variables (columns). Nevertheless, a solution consists of only a few of these variables. To create only useful variables, we use the CG algorithm [99]. As a reminder, CG is used for solving large-scale linear programs with the help of a back-and-forth resolution of a *master problem* and *pricing problems*. Beginning from a *reduced* master problem, the master problem gives dual values to the pricing problems, and the pricing problems give improving columns to the master problem. For every service k, we have to create an embedding γ among all possible embeddings Γ_k.

Master Problem

Only the set of variables $z_{k\gamma} \in \mathbb{N}$ is needed to represent the number of services k allocated on the embedding γ. Every embedding γ is determined by the amount of bandwidth it requires from every link e, as $\delta_e(\gamma)$, and the number of CPUs required from every node v, as $\theta_v(\gamma)$.

For every service $k \in K$, we limit the number of embedded with the constraints

$$\sum_{\gamma \in \Gamma_k} z_{k\gamma} \leq n_k, \tag{5.1a}$$

where n_k is the number of requests requiring the same service k.

For each node $v \in V$, we define its capacity constraints as:

$$\sum_{k \in K} \sum_{\gamma \in \Gamma_k} \theta_v(\gamma) z_{k\gamma} \leq C_v. \tag{5.1b}$$

And for each link $e \in E$, we define its capacity constraints as:

$$\sum_{k \in K} \sum_{\gamma \in \Gamma_k} \delta_e(\gamma) z_{k\gamma} \leq B_e, \tag{5.1c}$$

The objective function can be written as:

$$\max \sum_{k \in K} \sum_{\gamma \in \Gamma_k} z_{k\gamma}. \tag{5.1d}$$

Pricing Problems

Given a service $k \in K$, we formulate the corresponding embedding problem as an ILP.

— $x_e^l \in \{0, 1\}$ indicates if the virtual link $l \in L_k$ of the service is routed through the physical link $e \in E$.

— $y_v^f \in \{0, 1\}$ indicates if the VNF $f \in F_k$ of the service is instantiated in node $v \in V_f$.

The following set of constraints ensures that a VNF $\forall f \in F$ is embedded on a physical node:

$$\sum_{v \in V} y_v^f = 1. \tag{5.2a}$$

The following set of constraints ensure that a node $v \in V$ can host up to one VNF of the service:

$$\sum_{f \in F_k} y_v^f \leq 1 \tag{5.2b}$$

We ensure flow conservation for each node $v \in V$ and each VL $l = (f_i, f_j) \in L_k$ with:

$$\sum_{e \in \omega(v)} x_e^l - \sum_{e \in \omega^-(v)} x_e^l + y_v^{f_i} - y_v^{f_j} = 0. \tag{5.2c}$$

The overall latency of the service is ensured with:

$$\sum_{e \in E} D_e \sum_{l \in L_k} x_e^l \leq D^k, \tag{5.2d}$$

and the latency between each VL $l \in L_k$ with:

$$\sum_{e \in E} D_e x_e^l \leq D_k \tag{5.2e}$$

Pricing Objective Function

The pricing problems generate improving columns for the master problem. Improving and non-improving columns are different in their reduced costs, as the reduced cost of a variable represents the improvement of the objective function if the variable is present in

the solution. The pricing problem aims to find the columns with the best reduced cost. If all variables have a null reduced cost, then the CG algorithm has converged.

We can get a variable's reduced cost formula from the dual of the master problem. If we define by π the dual values corresponding to the constraints of the primal problem (and let the exponent define the corresponding constraint), the dual problem is formulated as:

$$\min \quad \sum_{k \in K} n_k \pi_k^{(5.1a)} + \sum_{e \in E} B_e \pi_e^{(5.1c)} + \sum_{v \in V} C_v \pi_v^{(5.1b)} \tag{5.3a}$$

$$s.t. \quad \pi^{(5.1a)} + \sum_{e \in E} \delta_e(\gamma) \pi_e^{(5.1c)} + \sum_{v \in V} \theta_v(\gamma) \pi_v^{(5.1b)} \geq 1$$

$$\forall k \in K, \forall \gamma \in \Gamma_k \tag{5.3b}$$

Columns in the master problem turn into constraints in the dual problem. Similarly to the CG algorithm, the row generation algorithm begins from a reduced problem and looks for unsatisfied constraints. Given a dual solution $\bar{\pi}$, the separation problem of the dual searches an embedding that violates constraints (5.3b), *i.e.,* any embedding γ such that

$$\bar{\pi}^{(5.1a)} + \sum_{e \in E} \delta_e(\gamma) \bar{\pi}_e^{(5.1c)} + \sum_{v \in V} \theta_v(\gamma) \bar{\pi}_v^{(5.1b)} < 1. \tag{5.4}$$

Returning to the embedding sub-problem for a given service k, defined by constraints (5.2), the objective function becomes

$$\min \sum_{e \in E} \bar{\pi}_e^{(5.1c)} \sum_{l \in L_k} B_l x_{el} + \sum_{v \in V} \sum_{f \in F_k} C_f \bar{\pi}_v^{(5.1b)} y_{vf}. \tag{5.5}$$

If the optimal value of this problem is strictly less than $1 - \pi_k^{(5.1a)}$, we know that adding the corresponding embedding into the master problem will improve the solution. Otherwise, no embedding can improve the master problem.

5.4 Proposed Solution

In the previous section, we addressed our placement problem using CG. We can obtain the global optimum (*i.e.,* the maximum feasible number of placed services) with CG in an offline placement scenario. But, when it comes to the online placement, since we do not know all the service requests in advance, we need to ensure that the placement of the current service request improves the possibility of accepting future service requests.

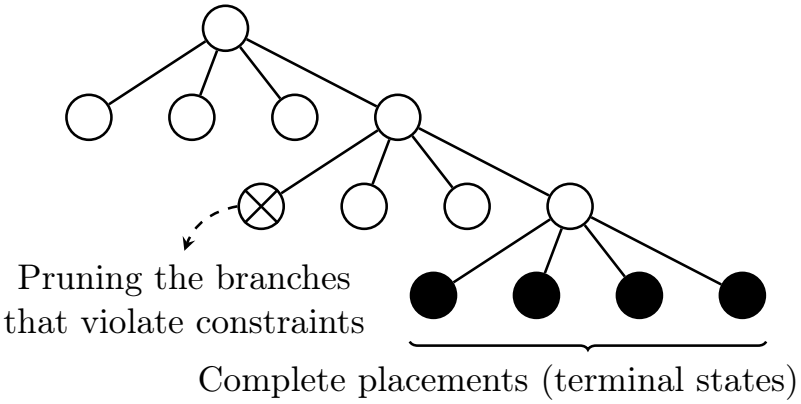


Figure 5.1 – A sample BnB search tree, placing a service with 3 VNFs over a network with 4 nodes

This can be done by optimizing a cost function which is consistent with our objective. For simplicity, we use the terms *network* and *service* to indicate the graphs of the Physical Network and the Virtual Service.

BnB is one of the most widely used paradigms of designing algorithms for solving optimization problems with exponential complexity. We have many constraints in the placement problem, which can be satisfied in BnB in an efficient way. Moreover, BnB allows a complete and optimal search over the solution space. Fig. 5.1 represents our BnB search tree.

We refer to a node of the search tree as a *state* to differentiate it from a network node. Each state carries placement information, as well as a complete image of the network with its resources and QoS metrics. The terminal states include a complete placement of a service over a network.

5.4.1 Expanding States

Beginning with the initial state of the tree, we choose a VNF from the service and create the associated sub-states for each network node that can supply the required resources (a parent-child relationship). A state includes a snapshot of the entire network after the placement of the VNF on the associated node. After the placement of a VNF, we place its connected VLs under the condition that their source and destination VNFs are already placed. To place a VL, we look for a shortest path from the node which places the source VNF to the node which places the destination VNF, taking into account the required bandwidth and latency. After placing a VLs, the network image of the state is updated

accordingly.

The search procedure is an iterative procedure involving a list of the states, which is called *fringe*, and it only contains the initial state at the beginning of the search.

At each iteration, we pop and expand the head state from the fringe, and we store the generated sub-states in the fringe. After expanding a state, we verify that the sub-states respect the constraints (concerning resources, latency, and anti-affinity). We only store the non-violating sub-states in the fringe (*i.e.,* others are pruned). The list of the states kept in the fringe specifies the traversal or the direction of the search.

To traverse the search tree in DFS, the sub-states of the current state are evaluated and sorted by the cost function, then they are sorted and pushed into the fringe (which is a stack in DFS) so that the head state (which will be popped for the expansion) has the minimum cost.

In Uniform-Cost Search (UCS) traversal, the sub-states of the current state are evaluated by the cost function and they are added to the fringe. The fringe in the UCS traversal is a sorted queue, and the head state has the minimum cost among all of the states. Thus, the UCS traversal is complete and optimal.

The search procedure continues the iterations until we reach a terminal state or we exceed a timeout (failure).

5.4.2 VNF Selection

At each state, we need to select a VNF to place it over the network. For this selection, we need an ordered list of the VNFs. To create this list, we make a BFS traversal over the service, beginning from the first VNF of the service which is used as the service entry (BFS traversal guarantees a connected sub-graph of the service on the states, representing a partial placement that can be evaluated against the constraints).

The depth of the corresponding state indicates the position of the VNF in the list of the VNFs to be selected for the placement.

The search tree grows by continuing to choose a VNF from the service and placing it on the network nodes, creating their associated sub-states.

The root state of the search tree is at depth 0, accordingly the states at depth 1 include the placement of the first VNF of the list, and the states of the final depth place the final VNF of the list. Therefore, if a service of N VNFs will be placed on a network of M nodes, we would have a search tree which will grow to the depth of N with a maximum branching factor of M.

Our approach does not enforce any limitation for sharing a VNF between services. If we need to share and reuse an already placed VNF (considering the fact that they must have the identical types), we only need to keep the branch that is associated with the node representing the location of the placed VNF, and prune the other branches.

5.4.3 Cost Function

The cost function evaluates the states to determine which one to expand at each iteration of the search procedure. It is used as a measurement to allow comparing different states. Since the states are located in different depths in the search tree and represent different partial or complete placements of the service, we need to ensure that the proposed cost function can compare all of the states, regardless of the number of VNFs and VLs that the state places.

5.4.4 Search Strategies

We define a search strategy including a cost function and a search traversal, that can be applied over our BnB formulation.

Latency Optimized Placement

We propose the LatUCS search strategy, which performs UCS traversal over a cost function which is defined as the sum of the latencies of the realized paths that place VLs. The idea is to investigate the effect of latency optimization on the objective of SA.

Random Placement

We propose the RanDFS search strategy, which performs a DFS traversal over a cost function that simply returns a random number. The idea is to investigate a worst-case random placement on our objective of SA.

Fair Placement

We propose several search strategies to realize a fair distribution of the node resources:

— VarUCS and VarDFS, which perform UCS and DFS traversals over a cost function which is defined as the variance of the available resources over all network nodes;

— RecUCS and RecDFS, which perform UCS and DFS traversals over a cost function which is defined as the average of the reciprocal function of the available resources over the network nodes that place VNFs ($\overline{\frac{1}{r+1}}$, +1 is added to prevent division by zero). We use the cost function proposed in [28], which tries to put a higher cost for placing on a network node which is short on resources.

5.5 Experimentation

Our implementations are made in Java, and we use the Gurobi solver. All of our experimentations are executed on a machine with a Core-i7 CPU and 8GB of RAM.

To evaluate our search strategies following our objective of SA, we begin by initializing the resources of the network nodes and links. Next, we repeatedly create a service request according to the specified parameters, and then we try to place it on the network using the specified strategy. If we accomplish placing the requested service, we carry out the placement by reserving its required resources on the network, and we begin a next iteration. If we fail to place the service request, we terminate the evaluation.

We perform our evaluations on the BT-Europe, BT-North-America and Grid-7x6 topologies (chosen from the dataset of Zoo Topology [101]), with respectively 24, 36 and 42 nodes, and 74, 152 and 142 links. We consider 10 units of CPU for each network node and 1 unit of latency for each network link, and daisy chain topology for the service graphs (representing a typical topology in SFCs). Each service contains between 3 to 5 VNFs with associated VLs, and each VNF requires a single unit of CPU. The user location is fixed at the network node *12* for BT-Europe, the node *34* for BT-North-America, and the node *0* for Grid-7x6. The E2E latency is calculated as the sum of the latencies of the paths that place the VLs, starting from the user location and passing through each VNF and returning to the user location.

5.5.1 Latency Range

Latency is a crucial constraint that should be set carefully. If a service requires the minimum feasible latency that can be provided by the network, we may not have many choices for the placement, and we are forced to place it as close as possible to the user location. Similarly, if a service requires a high enough E2E latency, so that we can place it almost wherever we want, regardless of the selected search strategy, we can place it on the

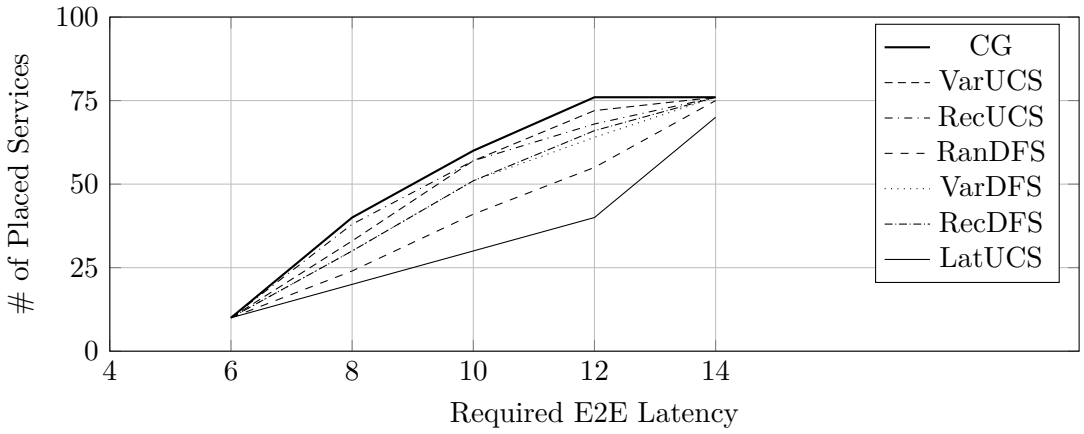


Figure 5.2 – Number of placed services comprised of 3 VNFs requiring different E2E latencies

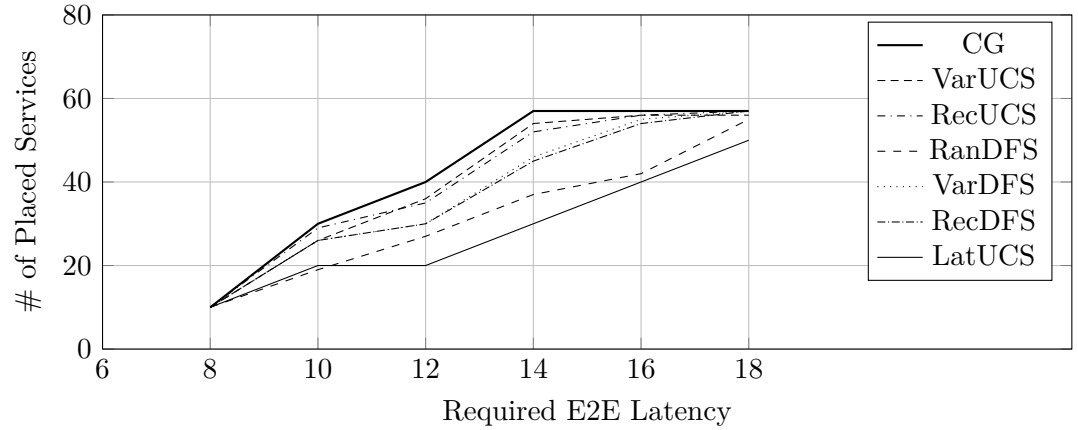
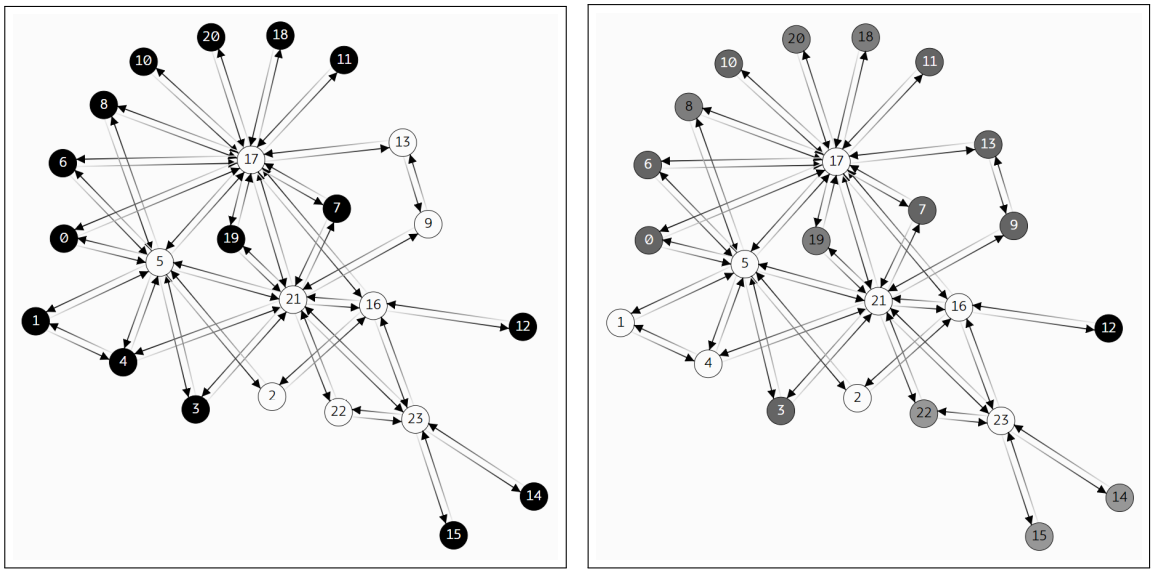


Figure 5.3 – Number of placed services comprised of 4 VNFs requiring different E2E latencies

cloud, where we do not worry about limited resources.

5.5.2 Latency Optimization

Fig. 5.2 and Fig. 5.3 represent the number of placed services comprised of 3 and 4 VNFs requiring different E2E latencies over BT-Europe network. As shown in Fig. 5.2, all of the search strategies place almost the same number of services for the lowest feasible latency (6 units), and the high-enough latency (14 units). Considering the E2E latency of 8 to 12 (called *effective* range), LatUCS places the minimum number of services, and CG has the maximum placements. Note that the number of placed services found by the CG is on average almost twice (exactly 1.96 times, or 96% improvement) the number of



(a) (b)

Figure 5.4 – Remaining available resources at the end of the evaluation applying LatUCS in (a), and VarUCS (b)

placements by LatUCS. Even RanDFS can place more services than LatUCS by making random placements (1.31 times more on average).

Graph Representation of Latency Optimization Effect

Fig. 5.4 demonstrates the remaining available resources of the network nodes at the end of the evaluation on the BT-Europe network, placing services of 3 VNFs requiring 10 units of E2E latency. We achieved 30 placements using LatUCS (Fig. 5.4(a)), and 50 placements using VarUCS (Fig. 5.4(b)). In this figure, the network nodes with full capacity are shown in black, the ones with no resources in white, and the others in gray-scale proportionate to their percentage of available resources. As mentioned before, by optimizing latency in LatUCS, we drain the resources around the user location (fixed on node 12), making it impossible to place more than 30 services. By optimizing the variance of the available resources in VarUCS, we prioritize using the nodes that have plenty of resources and avoid using the nodes that are short on resources. Thus, we can reach out to the resources on the nodes that are more distant, which leads to more service placements.

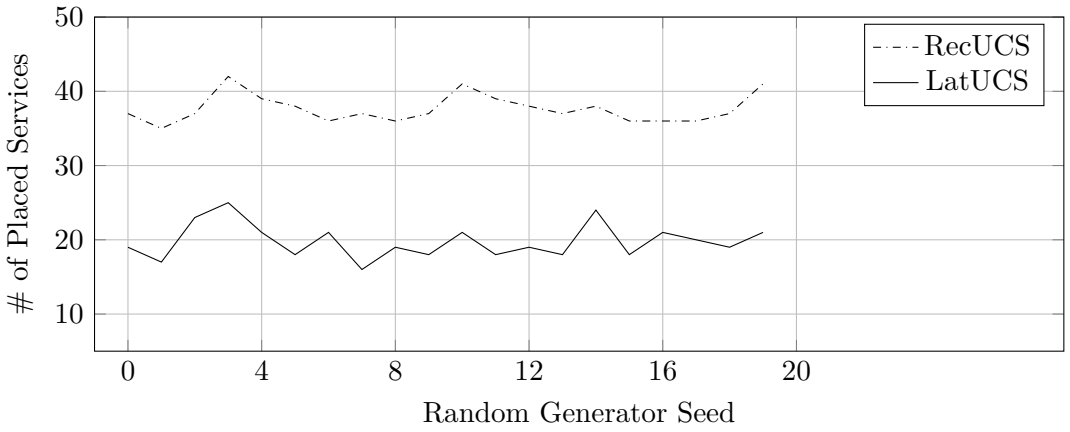


Figure 5.5 – Number of placed services in each random experiment

Random Service Requests

Until now, we had a constant size for the service requests in an evaluation loop. But, in the real world, the service requests can have different sizes, resource requirements, and latency requirements. To have realistic experiments, we need to create random services and repeat these experiments many times with multiple seeds of randomness. For every random experiment, we begin by creating a service of a random size in a range of [3, 5], with a random latency in the *effective* range based on its size.

Fig. 5.5 shows the number of placed services in each random experiment by applying RecUCS and LatUCS strategies. As it is shown in this figure, RecUCS can place almost 1.9 times on average more services by performing a fair placement, than LatUCS by performing latency optimization.

5.5.3 Fair Placement

Although VarUCS and RecUCS achieve almost the same results by placing 1.78 and 1.83 times more services than LatUCS on average, their execution times are considerably different. Fig. 5.6 and Fig. 5.7 show the average execution time of placing the services comprised of 3 and 4 VNFs requiring different E2E latencies over the BT-Europe network. As it is shown in Fig. 5.6, RecUCS can place the services 1.95 times faster on average than VarUCS. In addition, as it is shown in Fig. 5.7, the execution time of VarUCS grows significantly more by increasing the E2E latency. In RecUCS's case, the execution time decreases similarly to the DFS-based strategies, which shows the scalability of RecUCS.

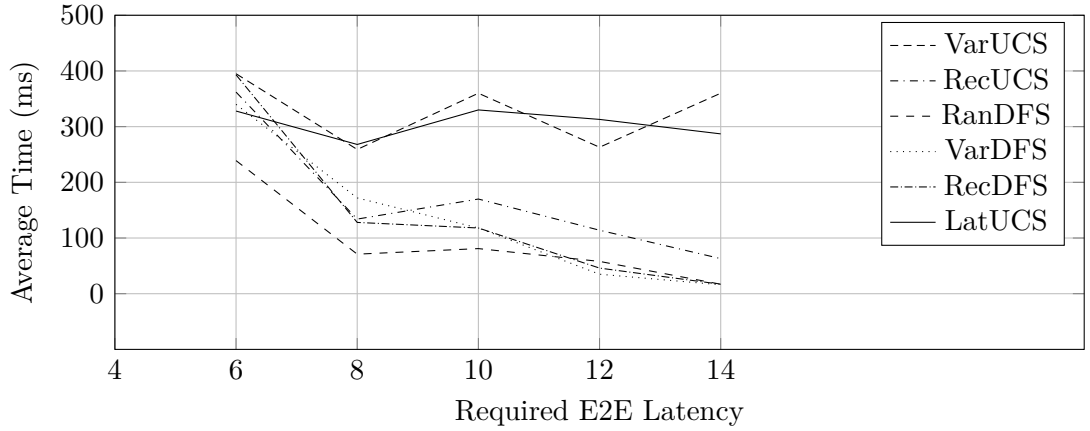


Figure 5.6 – Average time of placing services comprised of 3 VNFs requiring different E2E latencies

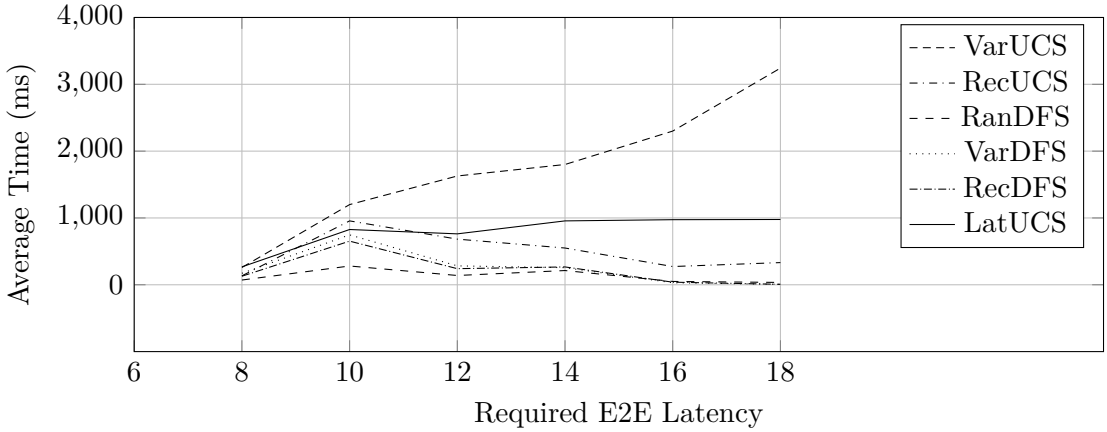


Figure 5.7 – Average time of placing services comprised of 4 VNFs requiring different E2E latencies

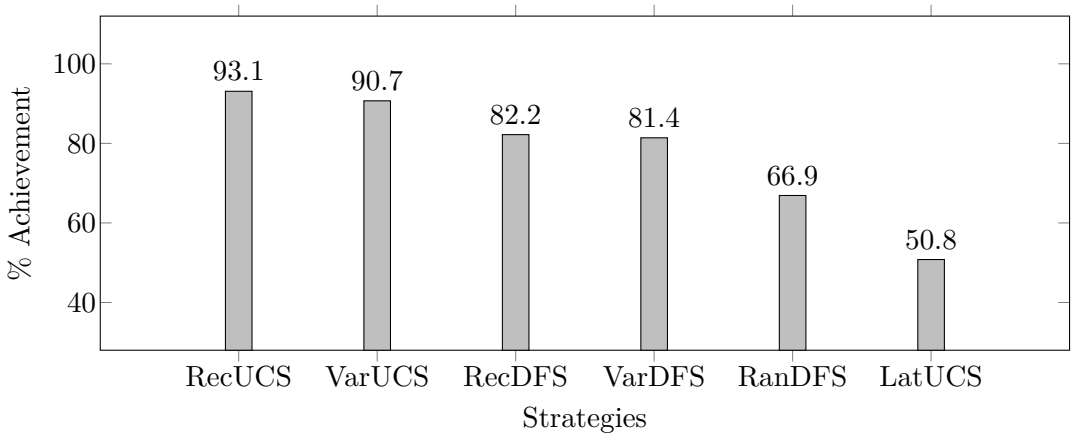


Figure 5.8 – Average achievement to the global optimal results by our online strategies

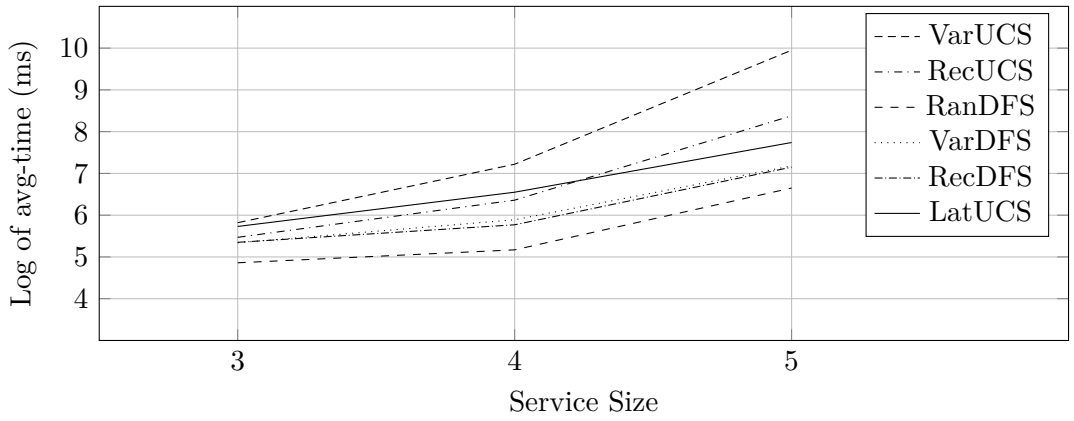


Figure 5.9 – Natural logarithm of the average placement time for placing different service sizes on BT-Europe network

5.5.4 Big Picture

Service Acceptance

Exhaustive evaluations were performed, considering 3 networks of BT-Europe, BT-North-America and Grid-7x6 with service requests of different sizes (3 to 5 VNFs), requiring an *effective* range of E2E latency, and user-location fixed on different network nodes. Fig. 5.8 shows the average percentage of the number of placements made by our proposed online placement strategies in comparison to our CG results which gives the maximum number of placements in an offline placement scenario.

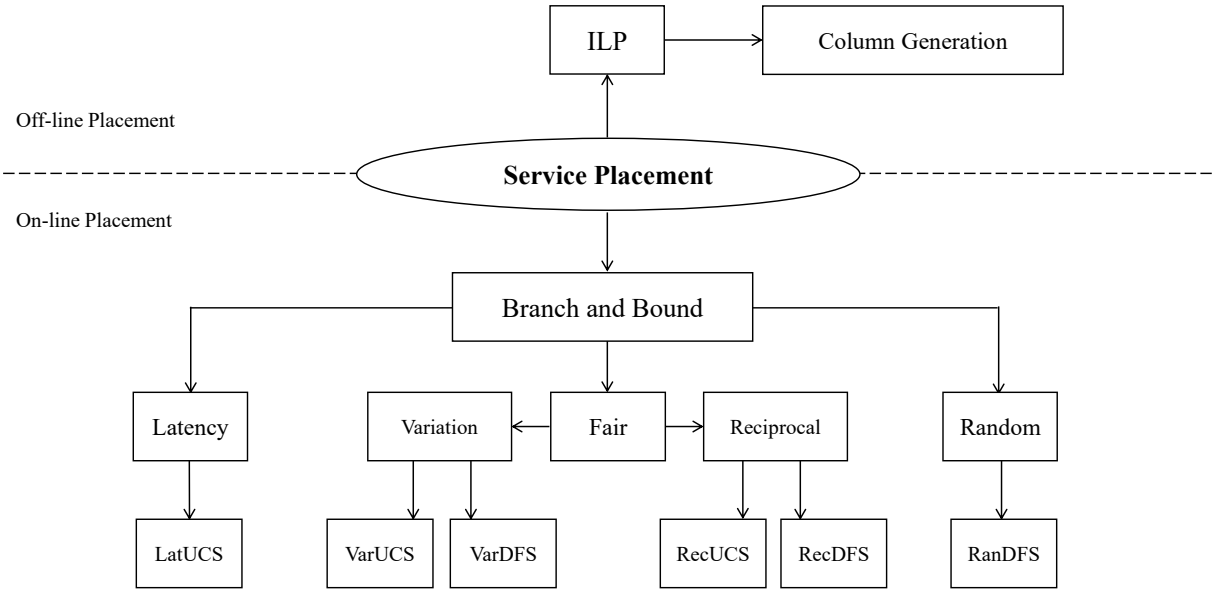


Figure 5.10 – Overview of the proposition

Execution Time

Since the execution time of our strategies grows exponentially by increasing the service size, we represent natural logarithm of the average placement execution time in Fig. 5.9. The complexity of our approach is clearly not polynomial (note that our problem is NP-Hard). The complexity of our approach depends on the growth of expanded states, which is determined by the search strategy. While the complexity of our strategies grows exponentially, they do not grow equally. If we can sort the growth of the complexity of our strategies, we can put VarUCS in the first place. RecUCS is placed between VarUCS and RecDFS & VarDFS (which have almost similar growth), and finally RanDFS.

5.6 Summary

In this chapter, we studied the online placement of the services with strict E2E latency and anti-affinity constraints considering the user location on edge networks with limited node resources, and we proposed several strategies to address this problem (fig. 5.10).

We proposed a CG approach to resolve the problem in an offline placement context to find the global optimum, and then we proposed an efficient online placement BnB approach, capable of achieving 93% of the global optimum.

We demonstrated that optimizing the bandwidth when we have limited network resources (which might sound a reasonable approach to maximize the service acceptance), it can have an opposite effect when our node resources are limited. In addition, we showed that a fair placement of the resources can be effective in maximizing the accepted services.

Chapter 6

Conclusion and Perspectives

6.1 Conclusion

The continuously evolving telecommunication technologies have led to higher expectations for 5G networks that can not be met simply by improving the technology of the current and the previous generation networks. Several new paradigms have been emerged like SDN and NFV, which demonstrated a high potential in creating flexible and scalable modern networks.

The placement of the Virtualized Network Services represents one of the most crucial steps in SDN and NFV. It tackles the allocation of physical resources for network services, demanding heterogeneous resources with various SLA constraints.

We provided a study of the related works to the placement of the network services, and we presented a literature review of the placement approaches in many NFV use-cases, following different objectives and constraints in Chapter 3.

Additionally, we explored the related works addressing the placement in specific contexts like limited bandwidth in Chapter 4. Although the VNF placement problem has been studied for many years, the need for an approach that could find a fair compromise between optimality and scalability still exists. We proposed a solution based on a BnB structure, adapting AI search strategies, particularly A^*, capable of finding optimal placements considerably fast. Substantial empirical analysis has been made, and the results confirm a considerable improvement (494% on average) in comparison to the popular Best-Fit algorithm.

Moreover, we studied the online placement of the services with strict E2E latency and anti-affinity constraints considering the user location on edge networks with limited node resources in Chapter 5. We proposed a CG approach to resolve the problem in an offline placement context to find the global optimum, and then we proposed an efficient online placement BnB approach, capable of achieving 93% of the global optimum.

6.2 Perspectives

The dynamic world of the telecommunications is so extensive, which we can only expect that hopefully we could spot, explore, find and finally put a few tiny pieces on this huge ever-changing puzzle. Our work was concentrated on the placement of the network services in NFV, maximizing the acceptance ratio (motivated by the revenue), considering the resources and QoS requirements.

To give a perspective of the possible future works to be explored and investigated, we divide them into two main categories.

6.2.1 Directly Related to Our Works

As a recall, we focused on our thesis over the algorithmic aspects of the placement of the network services in two general practical complementary problems that can occur regardless of the chosen use-case and constraints.

1. Limited bandwidth
2. Limited node resources

The algorithm that we proposed in the context of limited bandwidth is based on A^* algorithm that grows exponentially in memory on its execution. If the online placement imposes a time constraint, we only need to ensure that the search procedure will finish before using up the entire memory, depending on the scale of the network and the service. Otherwise, we need to see how to make our A^* algorithm more scalable. One of our suggestions is to use Simplified Memory Bounded A^* (SMA*).

In the limited node resources context, although we tried several solutions and cost functions, the only idea that looked promising was a fair spreading of the resources on the network. But, although the proposed algorithm can improve well the service acceptance, we think that there might be better approaches with less time complexity and more scalable.

The need for a unified algorithm, that is capable of finding suitable placements to maximize the service acceptance considering both limitations can be considered as another future direction. We might not be able to find such an algorithm, but at least finding a way to determine the limitation (whether it is related to the nodes or links) considering the actual substrate network and service, then adapting the placement algorithm accordingly, can be an interesting direction.

6.2.2 Taking a Step Back

During our thesis, we encountered several problems that seem interesting to work on.

Suppose that we have several users that are scattered over the network spreaded geographically over a large area, the problem is to determine how many and where to place the required network service. We could place a single service but it might not be able to satisfy the required QoS, or we could decide to place a separate replicate of the service for each of the users but it might be a waste of the resources. The question is what is the minimum number of required services and where to place them considering the QoS requirements and maybe the revenue and the cost. What just mentioned seems an interesting problem that to the best of our knowledge, it is not yet considered in the literature.

Another direction is to have a hierarchical view of the network service, for example by decomposing it into the VNFs and then VNFCs, and to scale only a portion of the service considering its graph hierarchy instead of replicating the whole service (in case that we need to scale, for example by adding new users or new network traffic). As an example, suppose that in our service graph, we have a VNF that is responsible for the authentication of the users, and we have another VNF for transcoding a video stream. An authenticator VNF might be able to authenticate many users once it is placed, but a transcoder might be able to serve only a few of the users considering its required resources since it could do a highly computation intensive task. Consequently, we might be able to scale only the transcoder (not the whole service) to serve the new users or new traffic. In general, we might be able to scale only a VNFC instead of a service that could be a big cluster of VNFs. This is another direction that seems interesting, and as far as we know, there are few works that try to decompose a network service and consider the scaling and placement of its elements. We tried to enter this area, but the priorities did not let us to make progress.

Last but not least, from the beginning of our thesis, we were curious to apply new techniques in ML for the placement problem. We tried to train a multi-layered neural network to predict the probability of the placement of a VNF over a network node by calculating it using the Law of Large Numbers (LLN) and Monte Carlo simulations. We even made implementations using PyTorch (as a ML library), but the results were not very promising. Although, we did not make a good progress in this direction, but it is always worth working on it by applying different problem formulations, and different learning techniques.

Chapter 7

Annex

7.1 Network Service Placer Tool

7.1.1 Introduction

In [24] as our first contribution, we proposed an online placement approach, based on BnB, allowing to apply various AI search strategies (especially A*). We demonstrated how to obtain near-optimal and optimal results with a high degree of scalability. Inspecting the effectiveness of our approach, we have struggled to find baseline solutions to which we could compare our results. We built the *NSPlacer* tool, to be served as the mentioned baseline solution, and we made extensive evaluations to inspect the effectiveness of our proposed approach. We developed and added new capabilities in our tool progressively during our PhD, and made the majority of our evaluations with it.

In this section, we present our *NSPlacer* tool [1], which we developed for making evaluations and comparisons between placement algorithms (including our approach proposed in [23], [24]). Since the placement problem is encountered in many areas related to NFV, *NSPlacer* may facilitate its investigation for the research community. By being able to adjust the related parameters of the substrate network, the service requests, and the placement algorithms, our online tool provides results, not only to see how the topology, node/link resources and QoS requirements influence the placement problem, but also to be used as a comparison among different placement solutions.

In addition, it provides an infrastructure for the researchers to evaluate their own placement solutions. As long as the parameters are compatible, the researchers can compare their results with the results obtained from our tool manually, or directly connect the implementation of their placement algorithm to our tool to provide the results. In order to make the evaluations between different placement algorithms more convenient, we provide a network-based web-socket API which allows a custom placement algorithm to

1. accessible online on `https://nsplacer.labs.b-com.com/`

be connected over the network regardless of its programming language [2].

We decided to make our *NSPlacer* tool public, offering the community a common tool to which they could compare their solutions. We believe that this can improve the reproducibility of the research results by providing a common environment for evaluating and comparing the algorithms.

This section is organized as follows. Section 7.1.2 is a small reminder of the problem of services placement. Section 7.1.3 provides an understanding of the architectural solution and describes how the tool performs the placements, while Section 7.1.4 gives a brief overview of the problem' parameters as well as the supported placement algorithms. Section 7.1.5, provides an idea of the outcomes that can be achieved using the proposed platform. Finally, Section 7.1.6 contains a summary of this section.

7.1.2 Problem

As a reminder, the physical network, also called substrate network, is defined as a graph of nodes (representing the servers) and links (representing the connections), along with a set of available resources for both nodes and links and QoS metrics. Similarly, a service is defined as a graph of VNFs and VLs, requiring a subset of the resources and subject to QoS constraints. The placement problem deals with finding a mapping of the VNFs and the VLs of a service onto the network nodes and the paths (the sequence of network links), by satisfying the corresponding constraints (Figure 7.1).

7.1.3 Tool architecture and processing

Figure 7.2 presents a brief overview of the architecture of our tool, with the different layers and technologies considered. The user interacts with the tool's web interface via the standard REST API using JSON as the data exchange format. Our tool uses the general architecture of the Spring [112] framework, which decomposes the architecture into several layers, including presentation, controllers, services and data (data model and data access).

The evaluations performed by our tool start by initializing the set of nodes and links in the network with the maximum available resources, and then attempt to place services iteratively. In each iteration, a service request is created based on the specified input parameters. Then, an attempt is made to place the service on the network using the selected

2. For more information on how to connect to our tool please visit the NSPlacer-Connect section on `https://nsplacer.labs.b-com.com/`

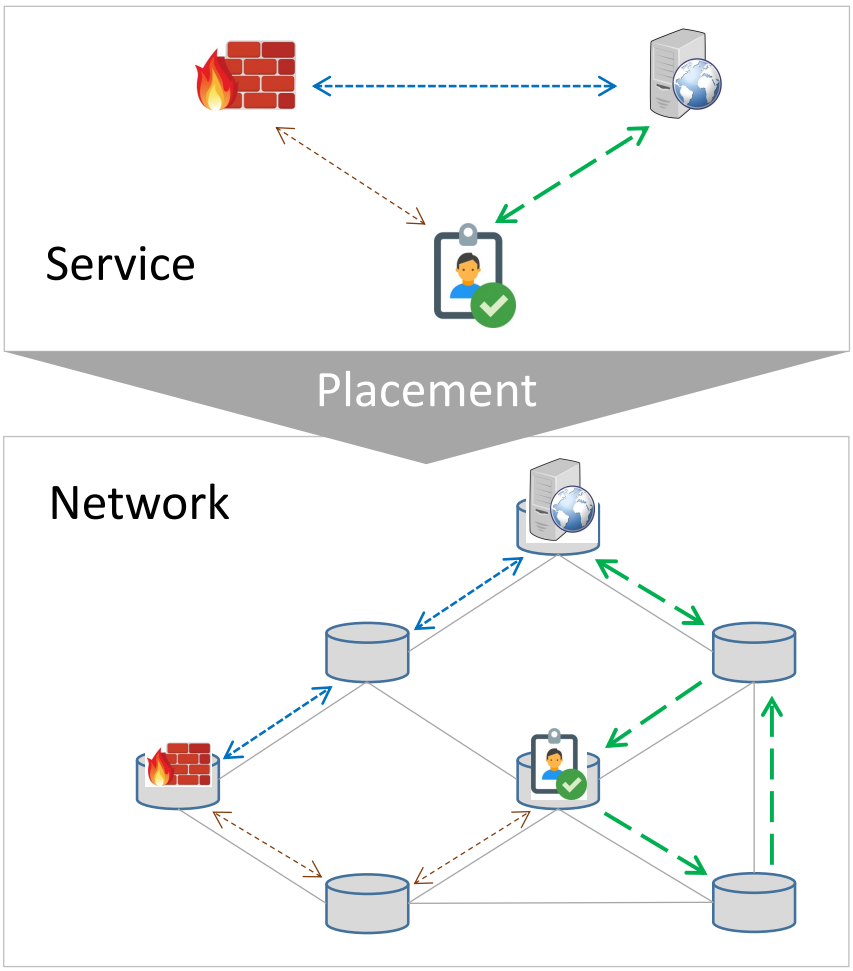


Figure 7.1 – Placement of a service request over the network

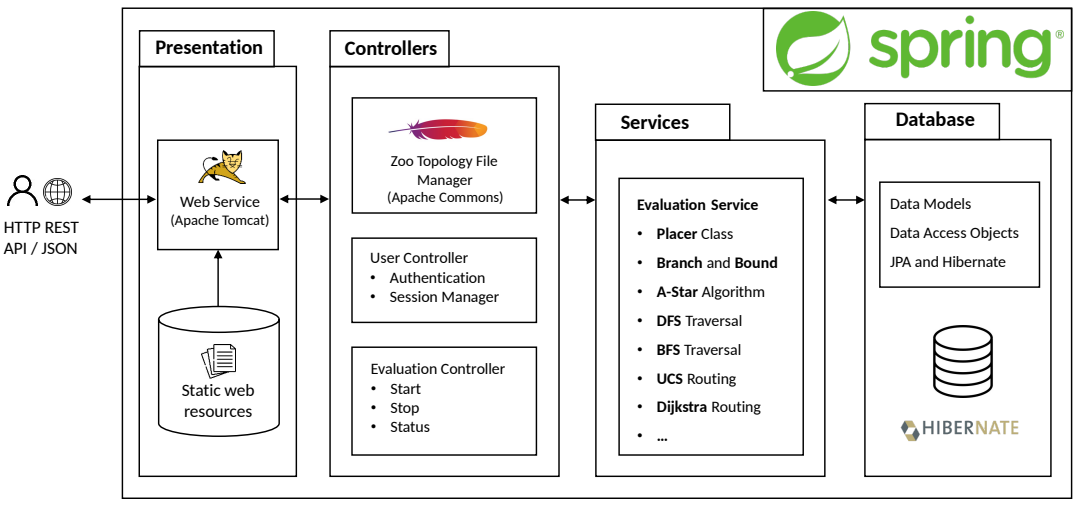


Figure 7.2 – General architecture

algorithm within the given time frame. If the algorithm successfully finds a placement, the placement is applied to the network by updating the remaining resources, before starting a new iteration. Otherwise, it completes the evaluation and reports the results. The results are shown at the end of the evaluation (Figure 7.4). They include the number of services placed, the execution time to place each service request in milliseconds, and the overall amount of resources remaining (CPU, storage, and bandwidth) at the end of the evaluation in percent. The exact placements of each service request are provided in the results section which can be easily exported.

7.1.4 Input/Output Parameters

In order to evaluate the influence of the inputs of the problem, the tool allows to choose the values of a large number of parameters classified in 3 categories.

Substrate Network

It is the physical network hosting the network services. Our tool accepts network topologies represented in the *Zoo Topology* [101] XML file format. New topologies can be downloaded from the zoo topology dataset, or defined as an XML file and uploaded into our tool. A graph visualization section is provided for the network topologies in order to be able to check the connectivity of the network visually. Currently, we consider *CPU* and *storage* as two resource types for each network node, along with the *bandwidth* and the *latency* for the network links (although latency is considered as a QoS metric, we use the term resource for unification). The Figure 7.3(a) shows the interface for setting the substrate network's parameters. At the beginning of the evaluation, these input parameters represent the initial capacity (maximum amount) of the resources for each network node and link.

Service Graph

It specifies the service request to be placed over the network. We propose different types of topologies, including Daisy-Chain, Ring and Star. The input parameter *size* specifies the number of VNFs. Each VNF requires several units of available resources from a network node, specified by the input parameters *CPU* and *storage*. Likewise, VLs require available resources from a network link defined by the input parameters *bandwidth* and *latency*. The interface is shown in Figure 7.3(b).

Placer

Our tool offers a set of input parameters to evaluate the algorithms as shown in Figure 7.3(c). In an online placement, we place one service at a time, as soon as it arrives, but within a *Timeout* for performing the placement. The *Termination* input parameter specifies whether the algorithm should search for better placements (*Best-Found*) until the timeout is reached or it should terminate as soon as it finds a first placement (*First-Found*). The parameter *Shuffle* is used to present randomness in the service placement algorithm. The *Routing* specifies the routing algorithm, either Dijkstra algorithm or Uniform Cost Search (UCS), to calculate the shortest paths. Last but not least, the input parameter *Algorithm* chooses the placement algorithm. These are either the predefined algorithms or the connected custom placement algorithms. It is worth reminding some of the predefined placement algorithms.

— *ABO:* ABO is one of the most interesting placement algorithms that we proposed, based on A* search algorithm over BnB to find optimal placement candidates by concentrating on the bandwidth usage optimization. A* is amongst the most popular AI informed search algorithms due to its completeness, optimality, and efficiency [100]. More details and the proof of the optimality of the ABO are provided in [24].

— *FABO:* Along with the optimality of ABO, increasing the chances of accepting the new service requests by avoiding network defragmentation is another important consideration. FABO is defined on top of ABO trying to address this issue by performing an optimal placement considering a fair distribution of the load over the network links.

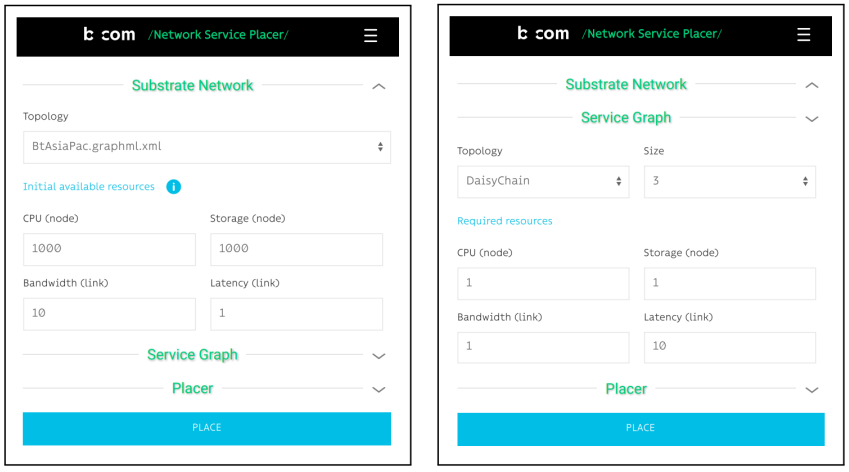
— *DBO:* DBO is another placement algorithm based on a DFS-traversal over BnB in order to quickly reach a near-optimal placement.

— *BF and EBF:* BF is one of the most well-known heuristic for service placement. By adapting BF over BnB, we could obtain other placement algorithm called EBF, which improves the results of BF remarkably.

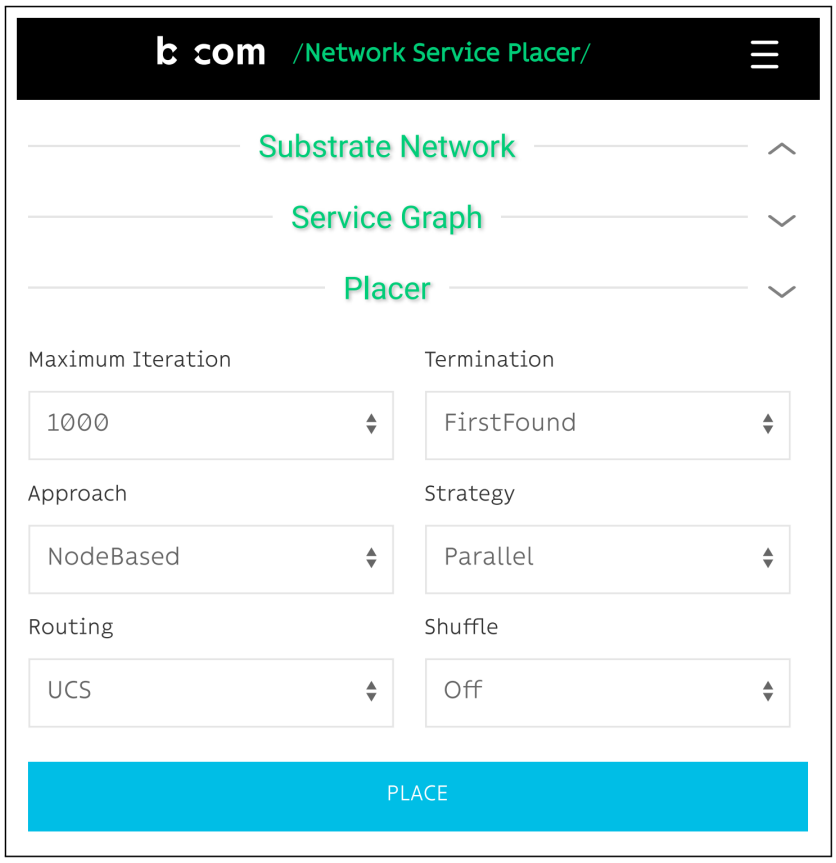
— *Parallel:* Parallel does not define a specific algorithm, but it executes FABO, ABO and DBO in parallel, and integrates them to obtain the best results.

7.1.5 Demonstration

To demonstrate a use-case of the tool and make comparisons between several placement algorithms, we conducted several following evaluations, and the results are summarized

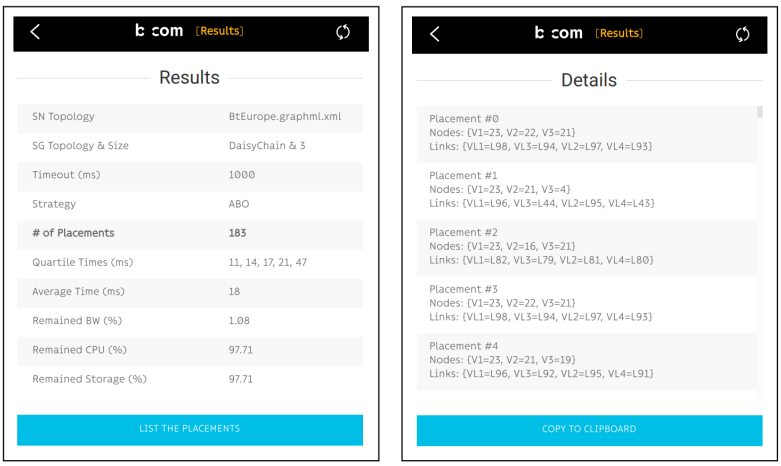


(a) (b)

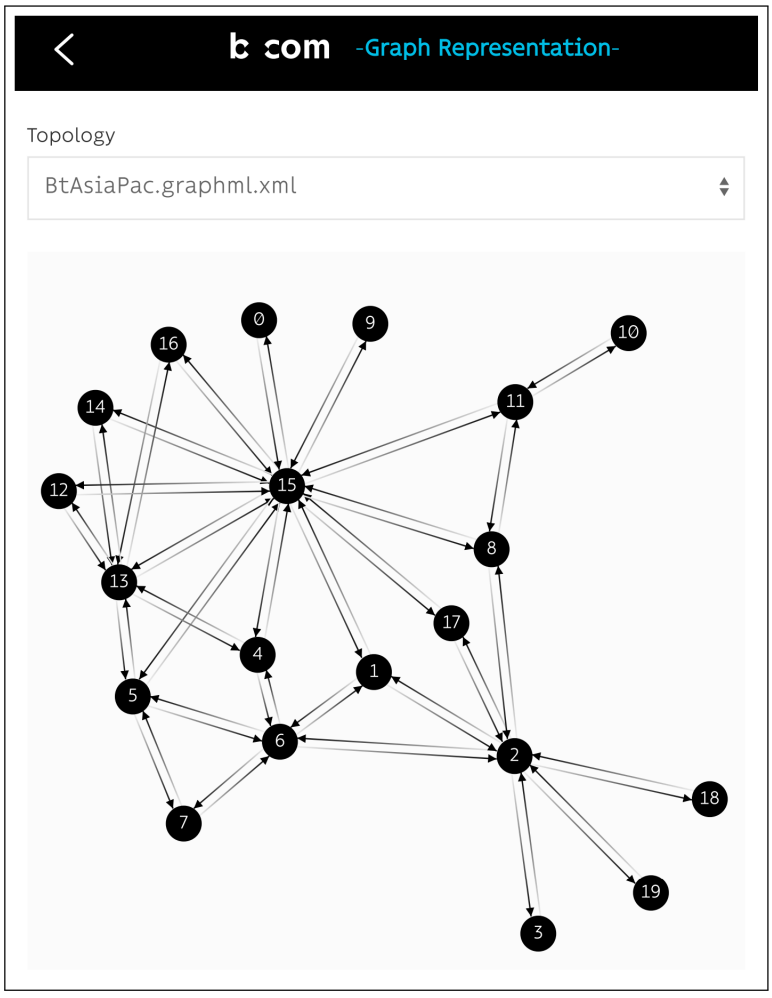


(c)

Figure 7.3 – (a) substrate network parameters, (b) service graph parameters, (c) placer parameters



(a) (b)



(c)

Figure 7.4 – (a) overall results, (b) placement details, (c) network graph representation

Table 7.1 – Evaluation results of placing services over BT-Europe

Algorithm	Services(#)	Time(ms)	Remained BW(%)
FABO	120	435	2.70
ABO	110	13	10
DBO	113	8	1.62
EBF	58	10	4.05
BF	26	9	59.59

in Table 7.1. We try to place service requests having 4 VNFs connected in Daisy-Chain topology (each VNF demanding 1 unit of CPU and 1 unit of storage, and each VL demanding 1 unit of bandwidth) over the network with Bt-Europe topology. We considered 1000 units of CPU and storage available for network nodes, and 10 units of bandwidth available for network links. We want to evaluate different placement algorithms. To do so, we set the input parameters, start the evaluation and collect the results (see figures 7.4(a) and 7.4(b)) for each execution and add the corresponding row in the table 7.1. Based on these results, we were able to show that FABO could place the highest number of services over the network but required considerably more time on average than the other algorithms. Although DBO could place a little more services than ABO, ABO could preserve much more remaining bandwidth at the end of the evaluation. In ABO, placed services uses less bandwidth on average than DBO, since ABO is guaranteed to be optimal, which is not the case for DBO. Although BF algorithm might perform well when we do not have significant constraints on bandwidth, it is obviously not a suitable algorithm when we have serious bandwidth constraints.

7.1.6 Summary

In this section, we presented our online tool “Network Service Placer”, as infrastructure for the research community to make evaluations with their own or predefined placement algorithms, not only to facilitate comparing placement algorithms but also to be able to track the influences of the topology, resources and QoS metrics on the placement algorithms. We believe that making our tool available to the community can contribute to the reproducibility of research results. Researchers will be able to compare their own solutions to common algorithms, or even integrate them into our tool to allow other researchers to evaluate theirs in return.

PUBLICATIONS

[1] M. Taghavian *et al.*, « An approach to network service placement reconciling optimality and scalability », *IEEE Transactions on Network and Service Management*, 2023

[2] M. Taghavian *et al.*, « An approach to network service placement using intelligent search strategies over branch-and-bound », in *2021 IEEE Global Communications Conference (GLOBECOM)*, 2021, pp. 1–6

[3] M. Taghavian *et al.*, « A fair approach to the online placement of the network services over the edge », in *2023 IEEE 19th International Conference on Network and Service Management (CNSM)*, 2023

[4] M. Taghavian *et al.*, « Nsplacer: a tool for evaluating service placement algorithms in post-5g networks », in *2022 IEEE 11th International Conference on Cloud Networking (CloudNet)*, 2022, pp. 252–256

[5] M. Taghavian *et al.*, « Placement de services virtualisés par des stratégies intelligentes pour l'algorithme de Branch and Bound », in *CORES 2022 – 7ème Rencontres Francophones sur la Conception de Protocoles, l'Évaluation de Performance et l'Expérimentation des Réseaux de Communication*, Saint-Rémy-Lès-Chevreuse, France, May 2022, pp. 1–4. [Online]. Available: `https://hal.science/hal-03663151`

BIBLIOGRAPHY

[1] U Cisco, « Cisco annual internet report (2018–2023) white paper », *Cisco: San Jose, CA, USA*, vol. 10, no. 1, pp. 1–35, 2020. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.pdf.

[2] *Mobile vs. desktop internet usage*, en. [Online]. Available: https://www.broadbandsearch.net/blog/mobile-desktop-internet-usage-statistics (visited on 12/10/2021).

[3] *What percentage of internet traffic is mobile?*, en. [Online]. Available: https://www.oberlo.com/statistics/mobile-internet-traffic (visited on 09/10/2022).

[4] *Mobile subscriptions forecast*, en. [Online]. Available: https://www.ericsson.com/en/reports-and-papers/mobility-report/dataforecasts/mobile-subscriptions-outlook (visited on 12/10/2021).

[5] *Setting the scene for 5g: opportunities and challenges*, en. [Online]. Available: http://handle.itu.int/11.1002/pub/811d7a5f-en (visited on 07/20/2023).

[6] F. A. Salaht, F. Desprez, and A. Lebre, « An overview of service placement problem in fog and edge computing », *ACM Computing Surveys (CSUR)*, vol. 53, no. 3, pp. 1–35, 2020.

[7] E. Haleplidis, K. Pentikousis, S. Denazis, J. H. Salim, D. Meyer, and O. Koufopavlou, « Software-defined networking (sdn): layers and architecture terminology », Tech. Rep., 2015.

[8] N. Feamster, J. Rexford, and E. Zegura, « The road to sdn: an intellectual history of programmable networks », *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 2, pp. 87–98, 2014.

[9] O. N. Foundation, *Sdn architecture*, 2014. [Online]. Available: https://opennetworking.org/wp-content/uploads/2013/02/TR_SDN_ARCH_1.0_06062014.pdf (visited on 08/01/2023).

[10] ETSI, *Network functions virtualisation: an introduction, benefits, enablers, challenges and call for action*, 2012. [Online]. Available: `https://portal.etsi.org/NFV/NFV_White_Paper.pdf` (visited on 08/01/2023).

[11] ETSI, *Network functions virtualisation: management and orchestration*, 2014. [Online]. Available: `https://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_NFV-MAN001v010101p.pdf` (visited on 08/01/2023).

[12] A. Leff and J. T. Rayfield, « Web-application development using the model/view/-controller design pattern », in *Proceedings fifth ieee international enterprise distributed object computing conference*, IEEE, 2001, pp. 118–127.

[13] N. Dragoni, S. Giallorenzo, A. L. Lafuente, M. Mazzara, F. Montesi, R. Mustafin, and L. Safina, « Microservices: yesterday, today, and tomorrow », *Present and ulterior software engineering*, pp. 195–216, 2017.

[14] B. B. Rad, H. J. Bhatti, and M. Ahmadi, « An introduction to docker and analysis of its performance », *International Journal of Computer Science and Network Security (IJCSNS)*, vol. 17, no. 3, p. 228, 2017.

[15] M. Luksa, *Kubernetes in action*. Simon and Schuster, 2017.

[16] ETSI, *Open-source mano*. [Online]. Available: `https://osm.etsi.org/` (visited on 08/01/2023).

[17] L. Foundation, *Open network automation platform*. [Online]. Available: `https://www.onap.org/` (visited on 08/01/2023).

[18] J. Medved, R. Varga, A. Tkacik, and K. Gray, « Opendaylight: towards a model-driven sdn controller architecture », in *Proceeding of IEEE international symposium on a world of wireless, mobile and multimedia networks 2014*, IEEE, 2014, pp. 1–6.

[19] N. Nikaein, M. K. Marina, S. Manickam, A. Dawson, R. Knopp, and C. Bonnet, « Openairinterface: a flexible platform for 5g research », *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 5, pp. 33–38, 2014.

[20] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, « Openflow: enabling innovation in campus networks », *ACM SIGCOMM computer communication review*, vol. 38, no. 2, pp. 69–74, 2008.

[21] Z. Chen, S. Zhang, C. Wang, Z. Qian, M. Xiao, J. Wu, and I. Jawhar, « A novel algorithm for nfv chain placement in edge computing environments », in *2018 IEEE Global Communications Conference (GLOBECOM)*, IEEE, 2018, pp. 1–6.

[22] G. Mirjalily and L. Zhiquan, « Optimal network function virtualization and service function chaining: a survey », *Chinese Journal of Electronics*, vol. 27, no. 4, pp. 704–717, 2018.

[23] M. Taghavian, Y. Hadjadj-Aoul, G. Texier, N. Huin, and P. Bertin, « An approach to network service placement reconciling optimality and scalability », *IEEE Transactions on Network and Service Management*, 2023.

[24] M. Taghavian, Y. Hadjadj-Aoul, G. Texier, and P. Bertin, « An approach to network service placement using intelligent search strategies over branch-and-bound », in *2021 IEEE Global Communications Conference (GLOBECOM)*, 2021, pp. 1–6.

[25] M. Taghavian, Y. Hadjadj-Aoul, G. Texier, and P. Bertin, « Nsplacer: a tool for evaluating service placement algorithms in post-5g networks », in *2022 IEEE 11th International Conference on Cloud Networking (CloudNet)*, 2022, pp. 252–256.

[26] M. Taghavian, Y. Hadjadj-Aoul, G. Texier, and P. Bertin, « Placement de services virtualisés par des stratégies intelligentes pour l'algorithme de Branch and Bound », in *CORES 2022 – 7ème Rencontres Francophones sur la Conception de Protocoles, l'Évaluation de Performance et l'Expérimentation des Réseaux de Communication*, Saint-Rémy-Lès-Chevreuse, France, May 2022, pp. 1–4. [Online]. Available: `https://hal.science/hal-03663151`.

[27] M. Taghavian, Y. Hadjadj-Aoul, G. Texier, and P. Bertin, « A fair approach to the online placement of the network services over the edge », in *2023 IEEE 19th International Conference on Network and Service Management (CNSM)*, 2023.

[28] C. Morin, G. Texier, C. Caillouet, G. Desmangles, and C.-T. Phan, « Vnf placement algorithms to address the mono-and multi-tenant issues in edge and core networks », in *2019 IEEE 8th International Conference on Cloud Networking (CloudNet)*, IEEE, 2019, pp. 1–6.

[29] S. Khebbache, M. Hadji, and D. Zeghlache, « Scalable and cost-efficient algorithms for vnf chaining and placement problem », in *2017 20th conference on innovations in clouds, internet and networks (ICIN)*, IEEE, 2017, pp. 92–99.

[30] P. Rodis and P. Papadimitriou, « Intelligent network service embedding using genetic algorithms », in *2021 IEEE Symposium on Computers and Communications (ISCC)*, IEEE, 2021, pp. 1–7.

[31] P. T. A. Quang, Y. Hadjadj-Aoul, and A. Outtagarts, « A deep reinforcement learning approach for vnf forwarding graph embedding », *IEEE Transactions on Network and Service Management*, vol. 16, no. 4, pp. 1318–1331, 2019.

[32] A. Rkhami, Y. Hadjadj-Aoul, and A. Outtagarts, « Learn to improve: a novel deep reinforcement learning approach for beyond 5g network slicing », in *2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC)*, IEEE, 2021, pp. 1–6.

[33] J. Pei, P. Hong, M. Pan, J. Liu, and J. Zhou, « Optimal vnf placement via deep reinforcement learning in sdn/nfv-enabled networks », *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 2, pp. 263–278, 2019.

[34] J. Sun, G. Huang, G. Sun, H. Yu, A. K. Sangaiah, and V. Chang, « A q-learning-based approach for deploying dynamic service function chains », *Symmetry*, vol. 10, no. 11, p. 646, 2018.

[35] Y. Yue, B. Cheng, M. Wang, B. Li, X. Liu, and J. Chen, « Throughput optimization and delay guarantee vnf placement for mapping sfc requests in nfv-enabled networks », *IEEE Transactions on Network and Service Management*, vol. 18, no. 4, pp. 4247–4262, 2021.

[36] R Guerzoni *et al.*, « Network functions virtualisation: an introduction, benefits, enablers, challenges and call for action, introductory white paper », in *SDN and OpenFlow World Congress*, vol. 1, 2012, pp. 5–7.

[37] ETSI, *European telecommunications standards institute, industry specification groups (isg) - nfv*, 2015. [Online]. Available: `http://www.etsi.org/technologies-clusters/technologies/nfv` (visited on 04/13/2021).

[38] ETSI, *Etsi group specifications on network function virtualization. 1st phase documents*, 2015. [Online]. Available: `http://docbox.etsi.org/ISG/NFV/Open/Published/` (visited on 04/13/2021).

[39] *Network functions virtualisation (nfv) use cases*, en. [Online]. Available: `https://docbox.etsi.org/ISG/NFV/Open/Publications_pdf/Specs-Reports/NFV%20001v1.3.1%20-%20GR%20-%20Use%20Cases.pdf` (visited on 03/31/2021).

[40] M. M. Tajiki, S. Salsano, L. Chiaraviglio, M. Shojafar, and B. Akbari, « Joint energy efficient and qos-aware path allocation and vnf placement for service function chaining », *IEEE Transactions on Network and Service Management*, vol. 16, no. 1, pp. 374–388, 2018.

[41] W. Yang and C. Fung, « A survey on security in network functions virtualization », in *2016 IEEE NetSoft Conference and Workshops (NetSoft)*, IEEE, 2016, pp. 15–19.

[42] M. Schöller, M. Stiemerling, A. Ripke, and R. Bless, « Resilient deployment of virtual network functions », in *2013 5Th international congress on ultra modern telecommunications and control systems and workshops (ICUMT)*, IEEE, 2013, pp. 208–214.

[43] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow, *et al.*, « Onos: towards an open, distributed sdn os », in *Proceedings of the third workshop on Hot topics in software defined networking*, 2014, pp. 1–6.

[44] O. Alhussein, P. T. Do, J. Li, Q. Ye, W. Shi, W. Zhuang, X. Shen, X. Li, and J. Rao, « Joint vnf placement and multicast traffic routing in 5g core networks », in *2018 IEEE Global Communications Conference (GLOBECOM)*, IEEE, 2018, pp. 1–6.

[45] T.-W. Kuo, B.-H. Liou, K. C.-J. Lin, and M.-J. Tsai, « Deploying chains of virtual network functions: on the relation between link and server usage », *IEEE/ACM Transactions On Networking*, vol. 26, no. 4, pp. 1562–1576, 2018.

[46] A. Mohammadkhan, S. Ghapani, G. Liu, W. Zhang, K. Ramakrishnan, and T. Wood, « Virtual function placement and traffic steering in flexible and dynamic software defined networks », in *The 21st IEEE International Workshop on Local and Metropolitan Area Networks*, IEEE, 2015, pp. 1–6.

[47] M. Mechtri, C. Ghribi, and D. Zeghlache, « A scalable algorithm for the placement of service function chains », *IEEE transactions on network and service management*, vol. 13, no. 3, pp. 533–546, 2016.

[48] B. Sonkoly, J. Czentye, M. Szalay, B. Németh, and L. Toka, « Survey on placement methods in the edge and beyond », *IEEE Communications Surveys & Tutorials*, vol. 23, no. 4, pp. 2590–2629, 2021.

[49] F. Schardong, I. Nunes, and A. Schaeffer-Filho, « Nfv resource allocation: a systematic review and taxonomy of vnf forwarding graph embedding », *Computer Networks*, vol. 185, p. 107 726, 2021.

[50] A. Alashaikh, E. Alanazi, and A. Al-Fuqaha, « A survey on the use of preferences for virtual machine placement in cloud data centers », *ACM Computing Surveys (CSUR)*, vol. 54, no. 5, pp. 1–39, 2021.

[51] X. Li and C. Qian, « A survey of network function placement », in *2016 13th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, IEEE, 2016, pp. 948–953.

[52] S. Demirci and S. Sagiroglu, « Optimal placement of virtual network functions in software defined networks: a survey », *Journal of Network and Computer Applications*, vol. 147, p. 102 424, 2019.

[53] O. A. Wahab, N. Kara, C. Edstrom, and Y. Lemieux, « Maple: a machine learning approach for efficient placement and adjustment of virtual network functions », *Journal of Network and Computer Applications*, vol. 142, pp. 37–50, 2019.

[54] M. Golkarifard, C. F. Chiasserini, F. Malandrino, and A. Movaghar, « Dynamic vnf placement, resource allocation and traffic routing in 5g », *Computer Networks*, vol. 188, p. 107 830, 2021.

[55] Z. Zhang, G. Wu, and H. Ren, « Multi-attribute-based qos-aware virtual network function placement and service chaining algorithms in smart cities », *Computers & Electrical Engineering*, vol. 96, p. 107 465, 2021.

[56] G. Yuan, Z. Xu, B. Yang, W. Liang, W. K. Chai, D. Tuncer, A. Galis, G. Pavlou, and G. Wu, « Fault tolerant placement of stateful vnfs and dynamic fault recovery in cloud networks », *Computer Networks*, vol. 166, p. 106 953, 2020.

[57] Q. Xu, D. Gao, T. Li, and H. Zhang, « Low latency security function chain embedding across multiple domains », *IEEE Access*, vol. 6, pp. 14 474–14 484, 2018.

[58] B. Németh, N. Molner, J. Martín-Pérez, C. J. Bernardos, A. De la Oliva, and B. Sonkoly, « Delay and reliability-constrained vnf placement on mobile and volatile 5g infrastructure », *IEEE Transactions on Mobile Computing*, vol. 21, no. 9, pp. 3150–3162, 2021.

[59] R. Zhang, Y. Chen, B. Dong, F. Tian, and Q. Zheng, « A genetic algorithm-based energy-efficient container placement strategy in caas », *IEEE Access*, vol. 7, pp. 121 360–121 373, 2019.

[60] A. Mseddi, W. Jaafar, H. Elbiaze, and W. Ajib, « Joint container placement and task provisioning in dynamic fog computing », *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 10 028–10 040, 2019.

[61] M. Gill and D. Singh, « Aco based container placement for caas in fog computing », *Procedia Computer Science*, vol. 167, pp. 760–768, 2020.

[62] A. M. Hafez, A. Abdelsamea, A. A. El-Moursy, S. M. Nassar, and M. B. E. Fayek, « Modified ant colony placement algorithm for containers », in *2020 15th International Conference on Computer Engineering and Systems (ICCES)*, 2020, pp. 1–6.

[63] L. Gu, D. Zeng, W. Li, S. Guo, A. Zomaya, and H. Jin, « Deep reinforcement learning based vnf management in geo-distributed edge computing », in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, IEEE, 2019, pp. 934–943.

[64] P. T. A. Quang, A. Bradai, K. D. Singh, and Y. Hadjadj-Aoul, « Multi-domain non-cooperative vnf-fg embedding: a deep reinforcement learning approach », in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, IEEE, 2019, pp. 886–891.

[65] S. I. Kim and H. S. Kim, « A research on dynamic service function chaining based on reinforcement learning using resource usage », in *2017 Ninth International Conference on Ubiquitous and Future Networks (ICUFN)*, IEEE, 2017, pp. 582–586.

[66] C. Pham, N. H. Tran, S. Ren, W. Saad, and C. S. Hong, « Traffic-aware and energy-efficient vnf placement for service chaining: joint sampling and matching approach », *IEEE Transactions on Services Computing*, vol. 13, no. 1, pp. 172–185, 2017.

[67] N. Kiran, X. Liu, S. Wang, and C. Yin, « Vnf placement and resource allocation in sdn/nfv-enabled mec networks », in *2020 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, IEEE, 2020, pp. 1–6.

[68] A. Alleg, T. Ahmed, M. Mosbah, R. Riggio, and R. Boutaba, « Delay-aware vnf placement and chaining based on a flexible resource allocation approach », in *2017 13th international conference on network and service management (CNSM)*, ieee, 2017, pp. 1–7.

[69] M. Leconte, G. S. Paschos, P. Mertikopoulos, and U. C. Kozat, « A resource allocation framework for network slicing », in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, IEEE, 2018, pp. 2177–2185.

[70] Z. Xiang, F. Gabriel, E. Urbano, G. T. Nguyen, M. Reisslein, and F. H. Fitzek, « Reducing latency in virtual machines: enabling tactile internet for human-machine co-working », *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 5, pp. 1098–1116, 2019.

[71] Z. Yan, J. Ge, Y. Wu, L. Li, and T. Li, « Automatic virtual network embedding: a deep reinforcement learning approach with graph convolutional networks », *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 6, pp. 1040–1057, 2020.

[72] Y. Mu, L. Wang, and J. Zhao, « Energy-efficient and interference-aware vnf placement with deep reinforcement learning », in *2021 IFIP Networking Conference (IFIP Networking)*, IEEE, 2021, pp. 1–9.

[73] R. Solozabal, J. Ceberio, A. Sanchoyerto, L. Zabala, B. Blanco, and F. Liberal, « Virtual network function placement optimization with deep reinforcement learning », *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 2, pp. 292–303, 2019.

[74] G. Kibalya, J. Serrat, J.-L. Gorricho, D. G. Bujjingo, J. Sserugunda, and P. Zhang, « A reinforcement learning approach for placement of stateful virtualized network functions », in *2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, IEEE, 2021, pp. 672–676.

[75] R. Behravesh, D. Harutyunyan, E. Coronado, and R. Riggio, « Time-sensitive mobile user association and sfc placement in mec-enabled 5g networks », *IEEE Transactions on Network and Service Management*, vol. 18, no. 3, pp. 3006–3020, 2021.

[76] J. J. A. Esteves, A. Boubendir, F. Guillemin, and P. Sens, « Location-based data model for optimized network slice placement », in *2020 6th IEEE Conference on Network Softwarization (NetSoft)*, IEEE, 2020, pp. 404–412.

[77] T. Kong, L. Wang, D. Ma, Z. Xu, Q. Yang, and K. Chen, « A secure container deployment strategy by genetic algorithm to defend against co-resident attacks in cloud computing », in *2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, 2019, pp. 1825–1832.

[78] D. Dwiardhika, T. Tachibana, *et al.*, « Virtual network embedding based on security level with vnf placement », *Security and Communication Networks*, vol. 2019, 2019.

[79] Q. Zhang, F. Liu, and C. Zeng, « Adaptive interference-aware VNF placement for service-customized 5G network slices », in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, IEEE, 2019, pp. 2449–2457.

[80] S. S. Shinde, D. Marabissi, and D. Tarchi, « A network operator-biased approach for multi-service network function placement in a 5g network slicing architecture », *Computer Networks*, vol. 201, p. 108 598, 2021.

[81] A. De Domenico, Y.-F. Liu, and W. Yu, « Optimal virtual network function deployment for 5g network slicing in a hybrid cloud infrastructure », *IEEE Transactions on Wireless Communications*, vol. 19, no. 12, pp. 7942–7956, 2020.

[82] S. T. Arzo, R. Bassoli, F. Granelli, and F. H. Fitzek, « Study of virtual network function placement in 5g cloud radio access network », *IEEE Transactions on Network and Service Management*, vol. 17, no. 4, pp. 2242–2259, 2020.

[83] J. Yusupov, A. Ksentini, G. Marchetto, and R. Sisto, « Multi-objective function splitting and placement of network slices in 5g mobile networks », in *2018 IEEE Conference on Standards for Communications and Networking (CSCN)*, IEEE, 2018, pp. 1–6.

[84] A. N. Toosi, R. Mahmud, Q. Chi, and R. Buyya, « Management and orchestration of network slices in 5g, fog, edge and clouds », *Fog and Edge Computing: Principles and Paradigms*, vol. 8, pp. 79–96, 2019.

[85] Y. L. Lee, J. Loo, T. C. Chuah, and L.-C. Wang, « Dynamic network slicing for multitenant heterogeneous cloud radio access networks », *IEEE Transactions on Wireless Communications*, vol. 17, no. 4, pp. 2146–2161, 2018.

[86] H. Halabian, « Distributed resource allocation optimization in 5g virtualized networks », *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 3, pp. 627–642, 2019.

[87] R. Su, D. Zhang, R. Venkatesan, Z. Gong, C. Li, F. Ding, F. Jiang, and Z. Zhu, « Resource allocation for network slicing in 5g telecommunication networks: a survey of principles and models », *IEEE Network*, vol. 33, no. 6, pp. 172–179, 2019.

[88] A. Dalgkitsis, P.-V. Mekikis, A. Antonopoulos, G. Kormentzas, and C. Verikoukis, « Dynamic resource aware vnf placement with deep reinforcement learning for 5g networks », in *GLOBECOM 2020-2020 IEEE Global Communications Conference*, IEEE, 2020, pp. 1–6.

[89] L. Yala, P. A. Frangoudis, and A. Ksentini, « Latency and availability driven vnf placement in a mec-nfv environment », in *2018 IEEE Global Communications Conference (GLOBECOM)*, IEEE, 2018, pp. 1–7.

[90] B. Blanco, I. Taboada, J. O. Fajardo, F. Liberal, *et al.*, « A robust optimization based energy-aware virtual network function placement proposal for small cell 5g networks with mobile edge computing capabilities », *Mobile Information Systems*, vol. 2017, 2017.

[91] Q. Zhang, L. Liu, C. Pu, Q. Dou, L. Wu, and W. Zhou, « A comparative study of containers and virtual machines in big data environment », in *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, IEEE, 2018, pp. 178–185.

[92] L. Lv, Y. Zhang, Y. Li, K. Xu, D. Wang, W. Wang, M. Li, X. Cao, and Q. Liang, « Communication-aware container placement and reassignment in large-scale internet data centers », *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 3, pp. 540–555, 2019.

[93] O. Oleghe, « Container placement and migration in edge computing: concept and scheduling models », *IEEE Access*, vol. 9, pp. 68 028–68 043, 2021.

[94] S. Arora and A. Ksentini, « Dynamic resource allocation and placement of cloud native network services », in *ICC 2021-IEEE International Conference on Communications*, IEEE, 2021, pp. 1–6.

[95] E. H. Bourhim, H. Elbiaze, and M. Dieye, « Inter-container communication aware container placement in fog computing », in *2019 15th International Conference on Network and Service Management (CNSM)*, 2019, pp. 1–6.

[96] A. Al-Moalmi, J. Luo, A. Salah, K. Li, and L. Yin, « A whale optimization system for energy-efficient container placement in data centers », *Expert Systems with Applications*, vol. 164, p. 113 719, 2021, ISSN: 0957-4174.

[97] R. Zhou, Z. Li, and C. Wu, « An efficient online placement scheme for cloud container clusters », *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 5, pp. 1046–1058, 2019.

[98] T. Rausch, A. Rashed, and S. Dustdar, « Optimized container scheduling for data-intensive serverless edge computing », *Future Generation Computer Systems*, vol. 114, pp. 259–271, 2021, ISSN: 0167-739X.

[99] G. Desaulniers, J. Desrosiers, and M. M. Solomon, *Column generation*. Springer Science & Business Media, 2006.

[100] S. Russell and P. Norvig, *Artificial intelligence: a modern approach*. Prentice Hall, 2002.

[101] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, « The internet topology zoo », *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 9, pp. 1765–1775, 2011.

[102] P. Van Mieghem and F. A. Kuipers, « Concepts of exact qos routing algorithms », *IEEE/ACM Transactions on networking*, vol. 12, no. 5, pp. 851–864, 2004.

[103] N. Bouten, R. Mijumbi, J. Serrat, J. Famaey, S. Latré, and F. De Turck, « Semantically enhanced mapping algorithm for affinity-constrained service function chain requests », *IEEE Transactions on Network and Service Management*, vol. 14, no. 2, pp. 317–331, 2017.

[104] H. Zhu and C. Huang, « Availability-aware mobile edge application placement in 5g networks », in *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, 2017, pp. 1–6.

[105] C. Zeng, F. Liu, S. Chen, W. Jiang, and M. Li, « Demystifying the performance interference of co-located virtual network functions », in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, IEEE, 2018, pp. 765–773.

[106] W. K. Hastings, « Monte carlo sampling methods using markov chains and their applications », *Biometrika*, vol. 57, no. 1, pp. 97–109, 1970.

[107] P. Jin, X. Fei, Q. Zhang, F. Liu, and B. Li, « Latency-aware VNF chain deployment with efficient resource reuse at network edge », in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, IEEE, 2020, pp. 267–276.

[108] R. Gouareb, V. Friderikos, and A.-H. Aghvami, « Virtual network functions routing and placement for edge cloud latency minimization », *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 10, pp. 2346–2357, 2018.

[109] A. Leivadeas, G. Kesidis, M. Ibnkahla, and I. Lambadaris, « VNF placement optimization at the edge and cloud », *Future Internet*, vol. 11, no. 3, p. 69, 2019.

[110] Z. Xu, Z. Zhang, W. Liang, Q. Xia, O. Rana, and G. Wu, « Qos-aware VNF placement and service chaining for iot applications in multi-tier mobile edge networks », *ACM Transactions on Sensor Networks (TOSN)*, vol. 16, no. 3, pp. 1–27, 2020.

[111] T. Hickling, A. Zenati, N. Aouf, and P. Spencer, « Explainability in deep reinforcement learning: a review into current methods and applications », *ACM Computing Surveys*, vol. 56, no. 5, pp. 1–35, 2023.

[112] R. Johnson, J. Hoeller, A. Arendsen, and R Thomas, *Professional Java development with the Spring framework*. John Wiley & Sons, 2009.

COLLEGES SCIENCES
BRETAGNE POUR L'INGENIEUR
LOIRE ET LE NUMERIQUE

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Titre : Placement des VNFs dans les réseaux 5G en utilisant AI/ML

Mot clés : Virtualisation des fonctions du réseau, placement des services

Résumé : La transition inévitable des dispositifs matériels physiques vers des modules logiciels légers et réutilisables dans le cadre de la virtualisation des fonctions de réseau (NFV) offre d'innombrables possibilités tout en présentant plusieurs défis sans précédent. La satisfaction des attentes de la NFV dans les réseaux post-5G dépend fortement du placement efficace des services de réseau. L'allocation dynamique des ressources physiques pour les demandes de services en ligne exigeant des ressources hétérogènes selon des exigences de qualité de service spécifiques représente l'une des étapes les plus importantes de la conception NFV et un problème NP-Hard à résoudre. Cette complexité est rencontrée dans divers cas d'utilisation 5G NFV, qui sont liés au placement, à partir de VNF Forwarding-Graphs et Network Slicing, à la virtualisation du réseau central, CDN, IoT, etc., examinant de nombreux objectifs dans la littérature, allant des optimisations multi-objectifs basées sur les ressources à la consommation d'énergie, le coût des revenus, l'acceptation du service, la résilience, la disponibilité, la sécurité, etc. Dans cette thèse, nous nous intéressons au placement des services de réseau virtuel sur le réseau en essayant de maximiser le nombre de services acceptés en tenant compte de leurs exigences de qualité de service. Bien que le problème de placement des VNF soit étudié depuis de nombreuses années, le besoin d'une approche permettant de trouver un juste compromis entre l'optimalité et l'scalabilité existe toujours. Dans cette thèse, nous étudions plusieurs problèmes et défis dans le placement de services en réseau et proposons des solutions AL/ML en conséquence.

Title: VNF Placement in 5G Networks using AI/ML

Keywords: 5G, Network Function Virtualization, Service Placement, AI, ML

Abstract: The inevitable transition from physical hardware devices towards lightweight reusable software modules in Network Function Virtualization (NFV) introduces countless opportunities while presenting several unprecedented challenges. Satisfying NFV expectations in post-5G networks heavily depends on the efficient placement of network services. Dynamic allocation of physical resources for online service requests demanding heterogeneous resources under specific QoS requirements represents one of the most important steps in NFV design, and a NP-Hard problem to solve. This complexity is encountered in various 5G NFV use-cases, which are related to the placement, from VNF Forwarding-Graphs and Network Slicing, to the virtualization of the Core Network, CDN, IoT, etc., investigating numerous objectives in the literature, ranging from resources-based multi-objective optimizations to the energy consumption, cost of revenue, service acceptance, resiliency, availability, security, etc. In this thesis, we are interested in placing the virtual network services over the network by trying to maximize the number of accepted services considering their QoS requirements. Although the VNF placement problem has been studied for many years, the need for an approach that could find a fair compromise between optimality and scalability still exists. In this thesis, we study several problems and challenges in network service placement and propose AL/ML solutions accordingly.